TiDB Documentation

PingCAP Inc.

20230602

Table of Contents

1 Introduction

1.1 TiDB Introduction

TiDB ("Ti" stands for Titanium) is an open-source NewSQL database that supports Hybrid Transactional and Analytical Processing (HTAP) workloads. It is MySQL compatible and features horizontal scalability, strong consistency, and high availability.

TiDB can be deployed in either self-hosted or cloud environments. The following deployment options are officially supported by PingCAP:

- Ansible Deployment: This guide describes how to deploy TiDB using TiDB Ansible. It is strongly recommended for production deployment.
- Ansible Offline Deployment: If your environment has no access to the internet, you can follow this guide to see how to deploy a TiDB cluster offline using TiDB Ansible.
- Docker Deployment: This guide describes how to deploy TiDB using Docker.
- Kubernetes Deployment:

  You can use TiDB Operator to deploy TiDB on:

  - AWS EKS (Elastic Kubernetes Service)
  - GKE (Google Kubernetes Engine)
  - Google Cloud Shell
  - Alibaba Cloud ACK (Container Service for Kubernetes)

  Or deploy TiDB locally using:

  - kind
  - Minikube

- Binary Tarball Deployment: This guide describes how to deploy TiDB from a binary tarball in production. Guides for development and testing environments are also available.

1.1.1 Community provided blog posts & tutorials

The following list collects deployment guides and tutorials from the community. The content is subject to change by the contributors.

- How To Spin Up an HTAP Database in 5 Minutes with TiDB + TiSpark
- Developer install guide (single machine)
- TiDB Best Practices

Your contribution is also welcome! Feel free to open a pull request to add additional links.

1.1.2 Source code

Source code for all components of the TiDB platform is available on GitHub.

- TiDB
- TiKV
- PD
- TiSpark
- TiDB Operator

1.2 Benchmarks

1.2.1 How to Test TiDB Using Sysbench

In this test, Sysbench 1.0.14 and TiDB 3.0 Beta are used. It is recommended to use Sysbench 1.0 or later, which can be downloaded here.

1.2.1.1 Test environment

- Hardware recommendations
- The TiDB cluster is deployed according to the TiDB Deployment Guide. Suppose there are 3 servers in total. It is recommended to deploy 1 TiDB instance, 1 PD instance and 1 TiKV instance on each server. As for disk space, supposing that there are 32 tables and 10M rows of data on each table, it is recommended that the disk space where TiKV's data directory resides is larger than 512 GB.

The number of concurrent connections to a single TiDB cluster is recommended to be under 500. If you need to increase the concurrency pressure on the entire system, you can add TiDB instances to the cluster whose number depends on the pressure of the test.

IDC machines:

Type	Name
OS	Linux (CentOS 7.3.1611)
CPU	40 vCPUs, Intel® Xeon® CPU E5-2630 v4 @ 2.20GHz
RAM	128GB
DISK	Intel Optane SSD P4800X 375G * 1
NIC	10Gb Ethernet

1.2.1.2 Test plan

1.2.1.2.1 TiDB version information

Component	GitHash
TiDB	7a240818d19ae96e4165af9ea35df92466f59ce6
TiKV	e26ceadcdfe94fb6ff83b5abb614ea3115394bcd
PD	5e81548c3c1a1adab056d977e7767307a39ecb70

1.2.1.2.2 Cluster topology

Machine IP	Deployment instance
172.16.30.31	3*sysbench
172.16.30.33	1*tidb 1*pd 1*tikv
172.16.30.34	1*tidb 1*pd 1*tikv
172.16.30.35	1*tidb 1*pd 1*tikv

1.2.1.2.3 TiDB configuration

Higher log level means fewer logs to be printed and thus positively influences TiDB performance. Enable `prepared plan cache` in the TiDB configuration to lower the cost of optimizing execution plan. Specifically, you can add the following command in the TiDB configuration file:

```
[log]
level = "error"
[prepared-plan-cache]
enabled = true
```

1.2.1.2.4 TiKV configuration

Higher log level also means better performance for TiKV.

As TiKV is deployed in clusters, the Raft algorithm can guarantee that data is written into most of the nodes. Therefore, except the scenarios where data safety is extremely important, `sync-log` can be disabled in raftstore.

There are 2 Column Families (Default CF and Write CF) on TiKV cluster which are mainly used to store different types of data. For the Sysbench test, the Column Family that is used to import data has a constant proportion among TiDB clusters:

Default CF : Write CF = 4 : 1

Configuring the block cache of RocksDB on TiKV should be based on the machine's memory size, in order to make full use of the memory. To deploy a TiKV cluster on a 40GB virtual machine, it is suggested to configure the block cache as follows:

```
log-level = "error"
[raftstore]
sync-log = false
```

```
[rocksdb.defaultcf]
block-cache-size = "24GB"
[rocksdb.writecf]
block-cache-size = "6GB"
```

For TiDB 3.0 or later versions, you can also use the shared block cache to configure:

```
log-level = "error"
[raftstore]
sync-log = false
[storage.block-cache]
capacity = "30GB"
```

For more detailed information on TiKV performance tuning, see Tune TiKV Performance.

1.2.1.3 Test process

> **Note:**
>
> This test was performed without load balancing tools such as HAproxy. We run the Sysbench test on individual TiDB node and added the results up. The load balancing tools and the parameters of different versions might also impact the performance.

1.2.1.3.1 Sysbench configuration

This is an example of the Sysbench configuration file:

```
mysql-host={TIDB_HOST}
mysql-port=4000
mysql-user=root
mysql-password=password
mysql-db=sbtest
time=600
threads={8, 16, 32, 64, 128, 256}
report-interval=10
db-driver=mysql
```

The above parameters can be adjusted according to actual needs. Among them, `TIDB_HOST` is the IP address of the TiDB server (because we cannot include multiple addresses in the configuration file), `threads` is the number of concurrent connections in

the test, which can be adjusted in "8, 16, 32, 64, 128, 256". When importing data, it is recommended to set threads = 8 or 16. After adjusting `threads`, save the file named **config**.

See the following as a sample **config** file:

```
mysql-host=172.16.30.33
mysql-port=4000
mysql-user=root
mysql-password=password
mysql-db=sbtest
time=600
threads=16
report-interval=10
db-driver=mysql
```

1.2.1.3.2 Data import

Before importing the data, it is necessary to make some settings to TiDB. Execute the following command in MySQL client:

```
set global tidb_disable_txn_auto_retry = off;
```

Then exit the client. TiDB uses an optimistic transaction model that rolls back transactions when a concurrency conflict is found. Setting `tidb_disable_txn_auto_retry` to `off` turns on the automatic retry mechanism after meeting a transaction conflict, which can prevent Sysbench from quitting because of the transaction conflict error.

Restart MySQL client and execute the following SQL statement to create a database `sbtest`:

```
create database sbtest;
```

Adjust the order in which Sysbench scripts create indexes. Sysbench imports data in the order of "Build Table -> Insert Data -> Create Index", which takes more time for TiDB to import data. Users can adjust the order to speed up the import of data. Suppose that you use the Sysbench version 1.0.14. You can adjust the order in either of the following two ways:

- Download the modified oltp_common.lua file for TiDB and overwrite the `/usr/share/sysbench/oltp_common.lua` file with it.
- In `/usr/share/sysbench/oltp_common.lua`, move the lines 235-240 to be right behind the line 198.

> **Note:**
>
> This operation is optional and is only to save the time consumed by data import.

At the command line, enter the following command to start importing data. The config file is the one configured in the previous step:

```
sysbench --config-file=config oltp_point_select --tables=32 --table-size
    ↪ =10000000 prepare
```

1.2.1.3.3 Warming data and collecting statistics

To warm data, we load data from disk into the block cache of memory. The warmed data has significantly improved the overall performance of the system. It is recommended to warm data once after restarting the cluster.

Sysbench 1.0.14 does not provide data warming, so it must be done manually. If you are using a later version of Sysbench, you can use the data warming feature included in the tool itself.

Take a table sbtest7 in Sysbench as an example. Execute the following SQL to warming up data:

```
SELECT COUNT(pad) FROM sbtest7 USE INDEX (k_7);
```

Collecting statistics helps the optimizer choose a more accurate execution plan. The `analyze` command can be used to collect statistics on the table sbtest. Each table needs statistics.

```
ANALYZE TABLE sbtest7;
```

1.2.1.3.4 Point select test command

```
sysbench --config-file=config oltp_point_select --tables=32 --table-size
    ↪ =10000000 run
```

1.2.1.3.5 Update index test command

```
sysbench --config-file=config oltp_update_index --tables=32 --table-size
    ↪ =10000000 run
```

1.2.1.3.6 Read-only test command

```
sysbench --config-file=config oltp_read_only --tables=32 --table-size
    ↪ =10000000 run
```

1.2.1.4 Test results

32 tables are tested, each with 10M of data.

Sysbench test was carried on each of the tidb-servers. And the final result was a sum of all the results.

1.2.1.4.1 oltp_point_select

Type	Thread	TPS	QPS	avg.latency(ms)	.95.latency(ms)	max.latency(ms)
point_select	3*8	67502.55	67502.55	0.36	0.42	141.92
point_select	3*16	120141.84	120141.84	0.40	0.52	20.99
point_select	3*32	170142.92	170142.92	0.58	0.99	28.08
point_select	3*64	195218.54	195218.54	0.98	2.14	21.82
point_select	3*128	208189.53	208189.53	1.84	4.33	31.02

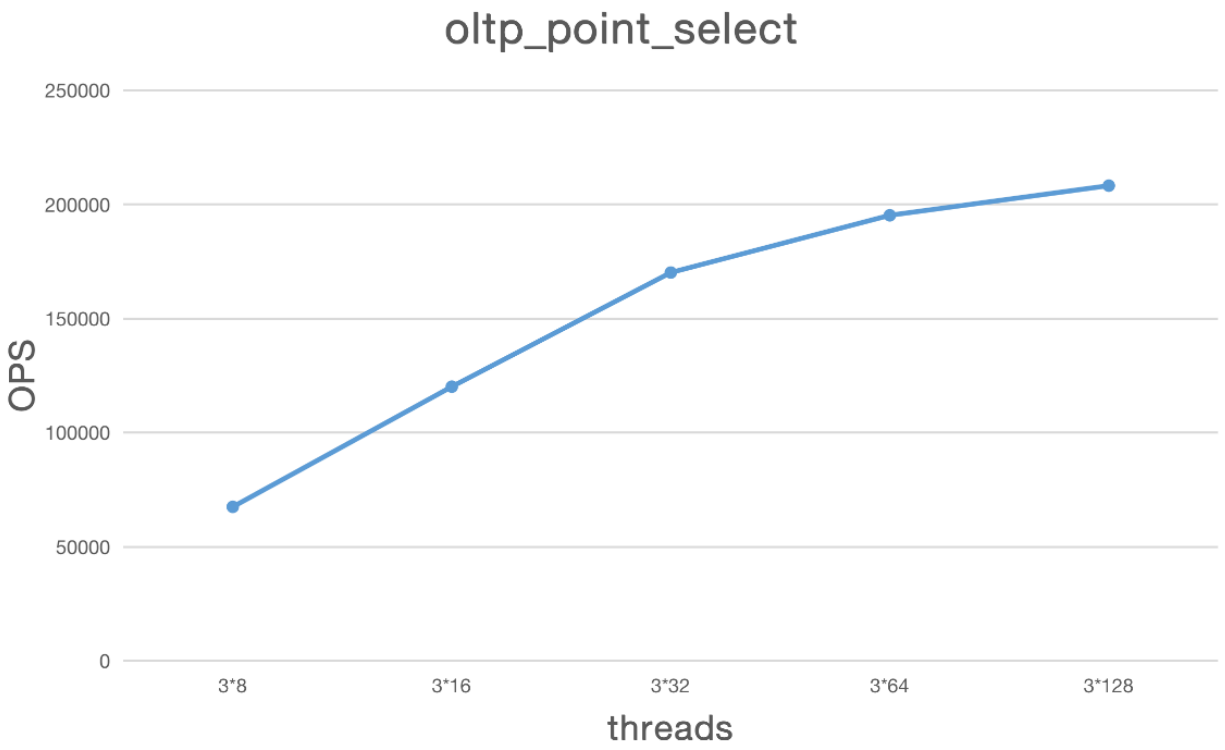

Figure 1: oltp_point_select

1.2.1.4.2 oltp_update_index

Type	Thread	TPS	QPS	avg.latency(ms)	.95.latency(ms)	max.latency(ms)
oltp_update_index	3*8	9668.98	9668.98	2.51	3.19	103.88
oltp_update_index	3*16	12834.99	12834.99	3.79	5.47	176.90
oltp_update_index	3*32	15955.77	15955.77	6.07	9.39	4787.14
oltp_update_index	3*64	18697.17	18697.17	10.34	17.63	4539.04
oltp_update_index	3*128	20446.81	20446.81	18.98	40.37	5394.75
oltp_update_index	3*256	23563.03	23563.03	32.86	78.60	5530.69

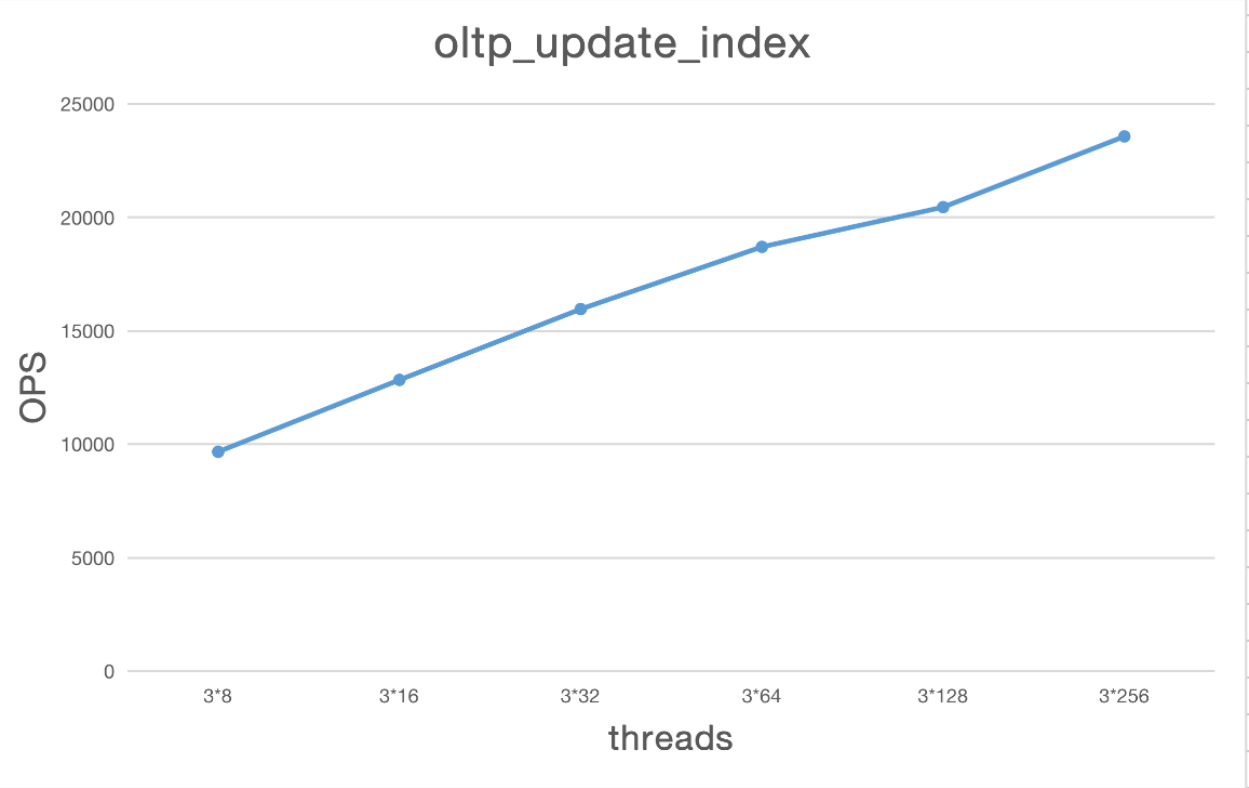

Figure 2: oltp_update_index

1.2.1.4.3 oltp_read_only

Type	Thread	TPS	QPS	avg.latency(ms)	.95.latency(ms)	max.latency(ms)
oltp_read_only	3*8	2411.00	38575.96	9.92	20.00	92.23
oltp_read_only	3*16	3873.53	61976.50	12.25	16.12	56.94
oltp_read_only	3*32	5066.88	81070.16	19.42	26.20	123.41
oltp_read_only	3*64	5466.36	87461.81	34.65	63.20	231.19
oltp_read_only	3*128	6684.16	106946.59	57.29	97.55	180.85

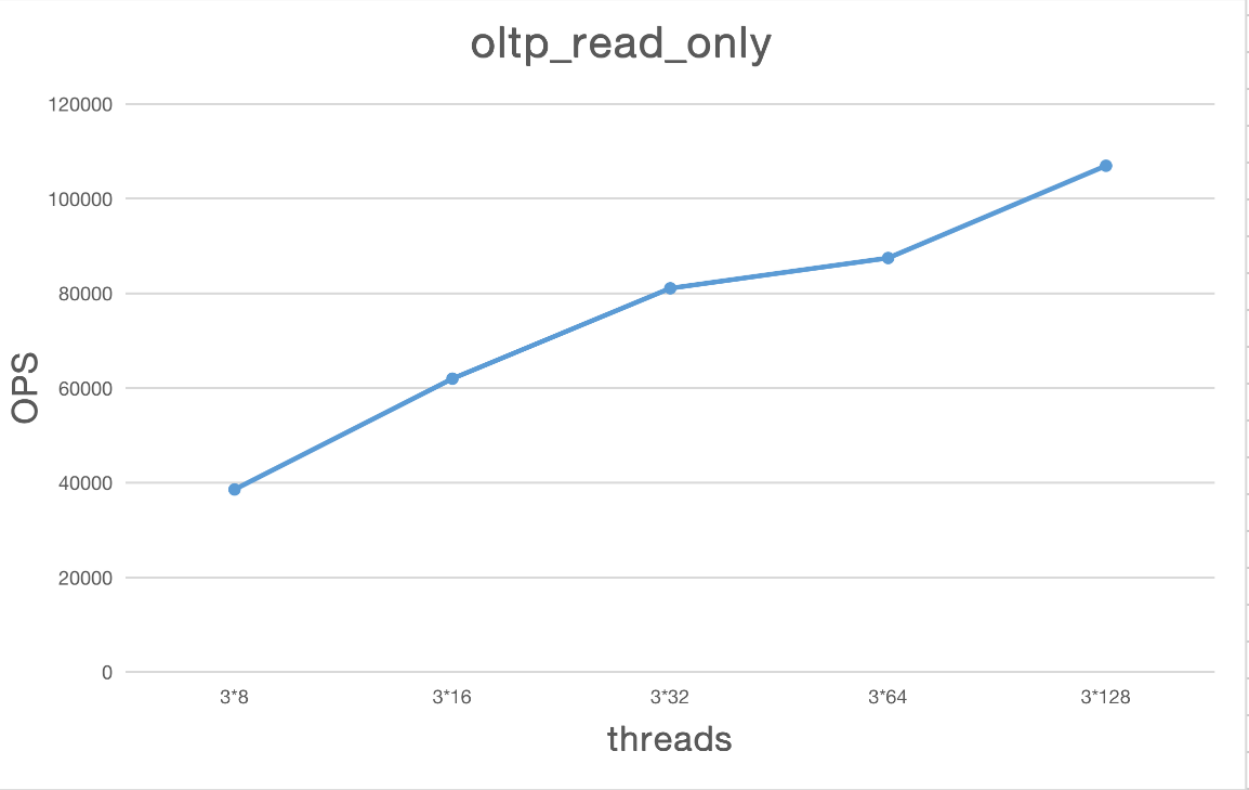

Figure 3: oltp_read_only

1.2.1.5 Common issues

1.2.1.5.1 TiDB and TiKV are both properly configured under high concurrency, why is the overall performance still low?

This issue often has things to do with the use of a proxy. You can add pressure on single TiDB server, sum each result up and compare the summed result with the result with proxy.

Take HAproxy as an example. The parameter `nbproc` can increase the number of processes it can start at most. Later versions of HAproxy also support `nbthread` and `cpu-map`. All of these can mitigate the negative impact of proxy use on performance.

1.2.1.5.2 Under high concurrency, why is the CPU utilization rate of TiKV still low?

Although the overall CPU utilization rate is low for TiKV, the CPU utilization rate of some modules in the cluster might be high.

The maximum concurrency limits for other modules on TiKV, such as storage readpool, coprocessor, and gRPC, can be adjusted through the TiKV configuration file.

The actual CPU usage can be observed through Grafana's TiKV Thread CPU monitor panel. If there is a bottleneck on the modules, it can be adjusted by increasing the concurrency of the modules.

1.2.1.5.3 Given that TiKV has not yet reached the CPU usage bottleneck under high concurrency, why is TiDB's CPU utilization rate still low?

CPU of NUMA architecture is used on some high-end equipment where cross-CPU access to remote memory will greatly reduce performance. By default, TiDB will use all CPUs of the server, and goroutine scheduling will inevitably lead to cross-CPU memory access.

Therefore, it is recommended to deploy n TiDBs (n is the number of NUMA CPUs) on the server of NUMA architecture, and meanwhile set the TiDB parameter `max-procs` to a value that is the same as the number of NUMA CPU cores.

1.2.2 How to Run TPC-C Test on TiDB

This document describes how to test TiDB using TPC-C.

TPC-C is an online transaction processing (OLTP) benchmark. It tests the OLTP system by using a commodity sales model that involves the following five transactions of different types:

- NewOrder
- Payment
- OrderStatus
- Delivery
- StockLevel

1.2.2.1 Prepare

Before testing, TPC-C Benchmark specifies the initial state of the database, which is the rule for data generation in the database. The `ITEM` table contains a fixed number of 100,000 items, while the number of warehouses can be adjusted. If there are W records in the `WAREHOUSE` table, then:

- The `STOCK` table has W * 100,000 records (Each warehouse corresponds to the stock data of 100,000 items)
- The `DISTRICT` table has W * 10 records (Each warehouse provides services to 10 districts)
- The `CUSTOMER` table has W * 10 * 3,000 records (Each district has 3,000 customers)
- The `HISTORY` table has W * 10 * 3,000 records (Each customer has one transaction history)
- The `ORDER` table has W * 10 * 3,000 records (Each district has 3,000 orders and the last 900 orders generated are added to the `NEW-ORDER` table. Each order randomly generates 5 ~ 15 ORDER-LINE records.

In this document, the testing uses 1,000 warehouses as an example to test TiDB.

TPC-C uses tpmC (transactions per minute) to measure the maximum qualified throughput (MQTh, Max Qualified Throughput). The transactions are the NewOrder transactions and the final unit of measure is the number of new orders processed per minute.

This testing uses the open-source BenchmarkSQL 5.0 as the TPC-C testing tool and adds the support for the MySQL protocol. You can download the testing program by using the following command:

```
git clone -b 5.0-mysql-support-opt-2.1 https://github.com/pingcap/
    ↪ benchmarksql.git
```

To install Java and Ant, run the following command (take CentOS as an example):

```
sudo yum install -y java ant
```

Enter the `benchmarksql` directory and run `ant` to build:

```
cd benchmarksql &&
ant
```

1.2.2.2 Deploy the TiDB cluster

For 1,000 warehouses, the TiDB cluster can be deployed on three machines, with one TiDB instance, one PD instance, and one TiKV instance on each machine.

For example, the hardware configuration is as follows:

Type	Name
OS	Linux (CentOS 7.3.1611)
CPU	40 vCPUs, Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
RAM	128GB
DISK	Optane 500GB SSD

1. Because this type of CPU has an NUMA architecture, it is recommended to bind the core using `taskset` and view the NUMA node using `lscpu`. For example:

```
NUMA node0 CPU(s):
    ↪ 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38
NUMA node1 CPU(s):
    ↪ 1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39
```

2. Start TiDB using the following command:

```
nohup taskset -c 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38
    ↪  bin/tidb-server &&
```

```
nohup taskset -c 1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39
    ↪ bin/tidb-server
```

3. You can deploy a HAproxy to balance the load of multiple TiDB nodes. It is recommended to configure `nbproc` as the number of CPU cores.

1.2.2.3 Edit the configuration

1.2.2.3.1 Configure TiDB

```
[log]
level = "error"

[performance]
### Sets the maximum number of CPU cores to use for a single TiDB instance
    ↪ according to NUMA configuration.
max-procs = 20

[prepared-plan-cache]
### Enables the prepared plan cache in TiDB configuration to reduce the
    ↪ overhead of optimizing your execution plan.
enabled = true
```

1.2.2.3.2 Configure TiKV

You can use the basic configuration at the beginning. Then after the test is run, you can adjust it based on the metrics on Grafana and the TiKV Tuning Instructions.

1.2.2.3.3 Configure BenchmarkSQL

Edit the `benchmarksql/run/props.mysql` file:

```
conn=jdbc:mysql://{HAPROXY-HOST}:{HAPROXY-PORT}/tpcc?useSSL=false&
    ↪ useServerPrepStmts=true&useConfigs=maxPerformance
warehouses=1000 # Uses 1,000 warehouses.
terminals=500 # Uses 500 terminals.
loadWorkers=32 # The number of concurrent workers that load data.
```

1.2.2.4 Load data

Loading data is usually the most time-consuming and problematic stage of the entire TPC-C test. This section provides the following four steps to load data.

First, use a MySQL client to connect to the TiDB server and run the following command:

```
create database tpcc;
```

Second, run the following BenchmarkSQL script in shell to create tables:

```
cd run && \
./runSQL.sh props.mysql sql.mysql/tableCreates.sql && \
./runSQL.sh props.mysql sql.mysql/indexCreates.sql
```

Third, use one of the following two ways to load data:

- Use BenchmarkSQL to load data directly
- Use TiDB Lightning to load data

1.2.2.4.1 Use BenchmarkSQL to load data directly

Run the following script to load data:

```
./runLoader.sh props.mysql
```

This process might last for several hours depending on the machine configuration.

1.2.2.4.2 Use TiDB Lightning to load data

The amount of loaded data increases as the number of warehouses increases. When you need to load more than 1000 warehouses of data, you can first use BenchmarkSQL to generate CSV files, and then quickly load the CSV files through TiDB Lightning (hereinafter referred to as Lightning). The CSV files can be reused multiple times, which saves the time required for each generation.

Follow the steps below to use TiDB Lightning to load data:

1. Modify the BenchmarkSQL configuration file.

   The CSV file of one warehouse requires 77 MB of disk space. To ensure sufficient disk space, add a line to the `benchmarksql/run/props.mysql` file:

   ```
   fileLocation=/home/user/csv/ # The absolute path of the directory where
       ↪ your CSV files are stored
   ```

   It is recommended that the CSV file names adhere to the naming rules in Lightning, that is, `{database}.{table}.csv`, because eventually you'll use Lightning to load data. Here you can modify the above configuration as follows:

   ```
   fileLocation=/home/user/csv/tpcc. # The absolute path of the directory
       ↪ where your CSV files are stored + the file name prefix (database)
   ```

   This will generate CSV files with a naming style such as `tpcc.bmsql_warehouse.csv`.

2. Generate the CSV file.

```
./runLoader.sh props.mysql
```

3. Use Lightning to load data.

   To load data using Lightning, see TiDB Lightning Deployment. The following steps introduce how to use TiDB Ansible to deploy Lightning and use Lightning to load data.

   1. Edit `inventory.ini`.

      It is recommended to manually specify the deployed IP address, the port, and the deployment directory to avoid anomalies caused by conflicts. For the disk space of `import_dir`, see TiDB Lightning Deployment. `data_source_dir` refers to the directory where the CSV files are stored as mentioned before.

```
[importer_server]
IS1 ansible_host=172.16.5.34 deploy_dir=/data2/is1
   ↪ tikv_importer_port=13323 import_dir=/data2/import
[lightning_server]
LS1 ansible_host=172.16.5.34 deploy_dir=/data2/ls1
   ↪ tidb_lightning_pprof_port=23323 data_source_dir=/home/user/
   ↪ csv
```

   2. Edit `conf/tidb-lightning.yml`.

```
mydumper:
    no-schema: true
    csv:
        separator: ','
        delimiter: ''
        header: false
        not-null: false
        'null': 'NULL'
        backslash-escape: true
        trim-last-separator: false
```

   3. Deploy Lightning and Importer.

```
ansible-playbook deploy.yml --tags=lightning
```

   4. Start Lightning and Importer.

- Log into the server where Lightning and Importer are deployed.
- Enter the deployment directory.
- Execute `scripts/start_importer.sh` under the Importer directory to start Importer.

- Execute `scripts/start_lightning.sh` under the Lightning directory to begin to load data.

Because you've used TiDB Ansible deployment method, you can see the loading progress of Lightning on the monitoring page, or check whether the loading process is completed through the log.

Fourth, after successfully loading data, you can run `sql.common/test.sql` to validate the correctness of the data. If all SQL statements return an empty result, then the data is correctly loaded.

1.2.2.5 Run the test

Run the following BenchmarkSQL test script:

```
nohup ./runBenchmark.sh props.mysql &> test.log &
```

After the execution is finished, view the result using `test.log`:

```
07:09:53,455 [Thread-351] INFO jTPCC : Term-00, Measured tpmC (NewOrders) =
    ↪ 77373.25
07:09:53,455 [Thread-351] INFO jTPCC : Term-00, Measured tpmTOTAL =
    ↪ 171959.88
07:09:53,455 [Thread-351] INFO jTPCC : Term-00, Session Start = 2019-03-21
    ↪ 07:07:52
07:09:53,456 [Thread-351] INFO jTPCC : Term-00, Session End = 2019-03-21
    ↪ 07:09:53
07:09:53,456 [Thread-351] INFO jTPCC : Term-00, Transaction Count = 345240
```

The value in the tpmC section is the testing result.

After the test completes, you can also run `sql.common/test.sql` to validate the correctness of the data. If all SQL statements return an empty result, then the testing of the data is correctly performed.

1.2.3 TiDB Sysbench Performance Test Report – v3.0 vs. v2.1

1.2.3.1 Test purpose

This test aims to compare the performance of TiDB 3.0 and TiDB 2.1 in the OLTP scenario.

1.2.3.2 Test version, time, and place

TiDB version: v3.0.0 vs. v2.1.13

Time: June, 2019

Place: Beijing

1.2.3.3 Test environment

This test runs on AWS EC2 and uses the CentOS-7.6.1810-Nitro (ami-028946f4cffc8b916) image. The components and types of instances are as follows:

Component	Instance type
PD	r5d.xlarge
TiKV	c5d.4xlarge
TiDB	c5.4xlarge

Sysbench version: 1.0.17

1.2.3.4 Test plan

Use Sysbench to import **16 tables, with 10,000,000 pieces of data in each table**. Start three sysbench to add pressure to three TiDB instances. The number of concurrent requests increases incrementally. A single concurrent test lasts 5 minutes.

Prepare data using the following command:

```
sysbench oltp_common \
    --threads=16 \
    --rand-type=uniform \
    --db-driver=mysql \
    --mysql-db=sbtest \
    --mysql-host=$tidb_host \
    --mysql-port=$tidb_port \
    --mysql-user=root \
    --mysql-password=password \
    prepare --tables=16 --table-size=10000000
```

Then test TiDB using the following command:

```
sysbench $testname \
    --threads=$threads \
    --time=300 \
    --report-interval=15 \
    --rand-type=uniform \
    --rand-seed=$RANDOM \
    --db-driver=mysql \
    --mysql-db=sbtest \
    --mysql-host=$tidb_host \
    --mysql-port=$tidb_port \
    --mysql-user=root \
    --mysql-password=password \
    run --tables=16 --table-size=10000000
```

1.2.3.4.1 TiDB version information

1.2.3.4.2 v3.0.0

Component	GitHash
TiDB	`8efbe62313e2c1c42fd76d35c6f020087eef22c2`
TiKV	`a467f410d235fa9c5b3c355e3b620f81d3ac0e0c`
PD	`70aaa5eee830e21068f1ba2d4c9bae59153e5ca3`

1.2.3.4.3 v2.1.13

Component	GitHash
TiDB	`6b5b1a6802f9b8f5a22d8aab24ac80729331e1bc`
TiKV	`b3cf3c8d642534ea6fa93d475a46da285cc6acbf`
PD	`886362ebfb26ef0834935afc57bcee8a39c88e54`

1.2.3.4.4 TiDB parameter configuration

Enable the prepared plan cache in both TiDB v2.1 and v3.0 (`point select` and `read write` are not enabled in v2.1 for optimization reasons):

```
[prepared-plan-cache]
enabled = true
```

Then configure global variables:

```
set global tidb_hashagg_final_concurrency=1;
set global tidb_hashagg_partial_concurrency=1;
set global tidb_disable_txn_auto_retry=0;
```

In addition, make the following configuration in v3.0:

```
[tikv-client]
max-batch-wait-time = 2000000
```

1.2.3.4.5 TiKV parameter configuration

Configure the global variable in both TiDB v2.1 and v3.0:

```
log-level = "error"
[readpool.storage]
normal-concurrency = 10
[server]
grpc-concurrency = 6
```

```
[rocksdb.defaultcf]
block-cache-size = "14GB"
[rocksdb.writecf]
block-cache-size = "8GB"
[rocksdb.lockcf]
block-cache-size = "1GB"
```

In addition, make the following configuration in v3.0:

```
[raftstore]
apply-pool-size = 3
store-pool-size = 3
```

1.2.3.4.6 Cluster topology

Machine IP	Deployment instance
172.31.8.8	3 * Sysbench
172.31.7.80, 172.31.5.163, 172.31.11.123	PD
172.31.4.172, 172.31.1.155, 172.31.9.210	TiKV
172.31.7.80, 172.31.5.163, 172.31.11.123	TiDB

1.2.3.5 Test result

1.2.3.5.1 `Point Select` test

v2.1:

Threads	QPS	95% latency(ms)
150	240304.06	1.61
300	276635.75	2.97
600	307838.06	5.18
900	323667.93	7.30
1200	330925.73	9.39
1500	336250.38	11.65

v3.0:

Threads	QPS	95% latency(ms)
150	334219.04	0.64
300	456444.86	1.10
600	512177.48	2.11
900	525945.13	3.13

Threads	QPS	95% latency(ms)
1200	534577.36	4.18
1500	533944.64	5.28

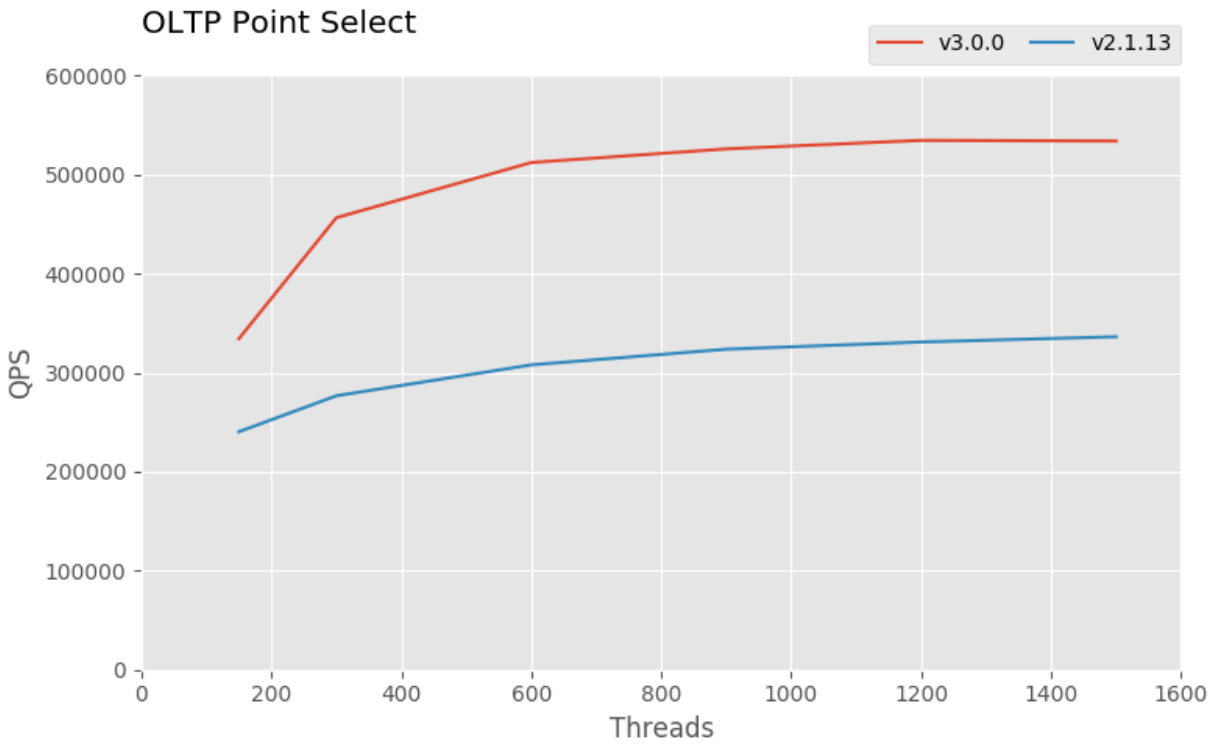

Figure 4: point select

1.2.3.5.2 `Update Non-Index test`

v2.1:

Threads	QPS	95% latency (ms)
150	21785.37	8.58
300	28979.27	13.70
600	34629.72	24.83
900	36410.06	43.39
1200	37174.15	62.19
1500	37408.88	87.56

v3.0:

Threads	QPS	95% latency (ms)
150	28045.75	6.67
300	39237.77	9.91
600	49536.56	16.71
900	55963.73	22.69
1200	59904.02	29.72
1500	62247.95	42.61

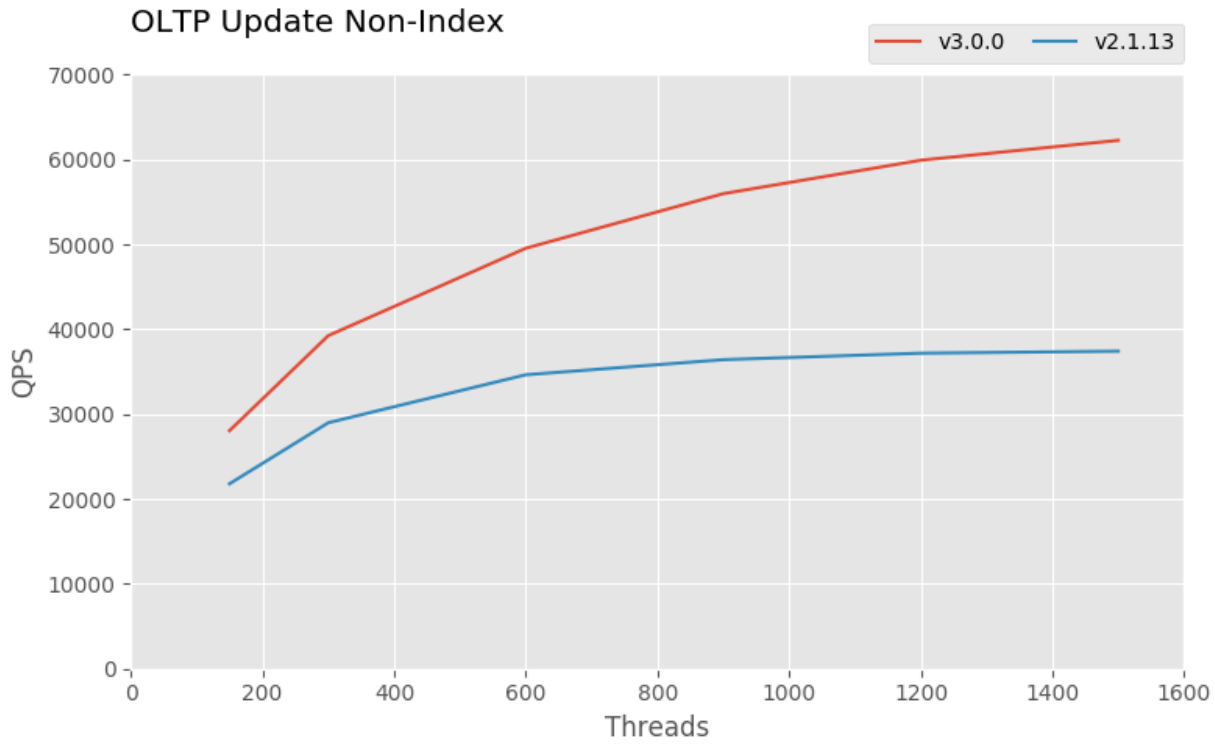

Figure 5: update non-index

1.2.3.5.3 `Update Index` test

v2.1:

Threads	QPS	95% latency(ms)
150	14378.24	13.22
300	16916.43	24.38
600	17636.11	57.87
900	17740.92	95.81
1200	17929.24	130.13
1500	18012.80	161.51

v3.0:

Threads	QPS	95% latency(ms)
150	19047.32	10.09
300	24467.64	16.71
600	28882.66	31.94
900	30298.41	57.87
1200	30419.40	92.42
1500	30643.55	125.52

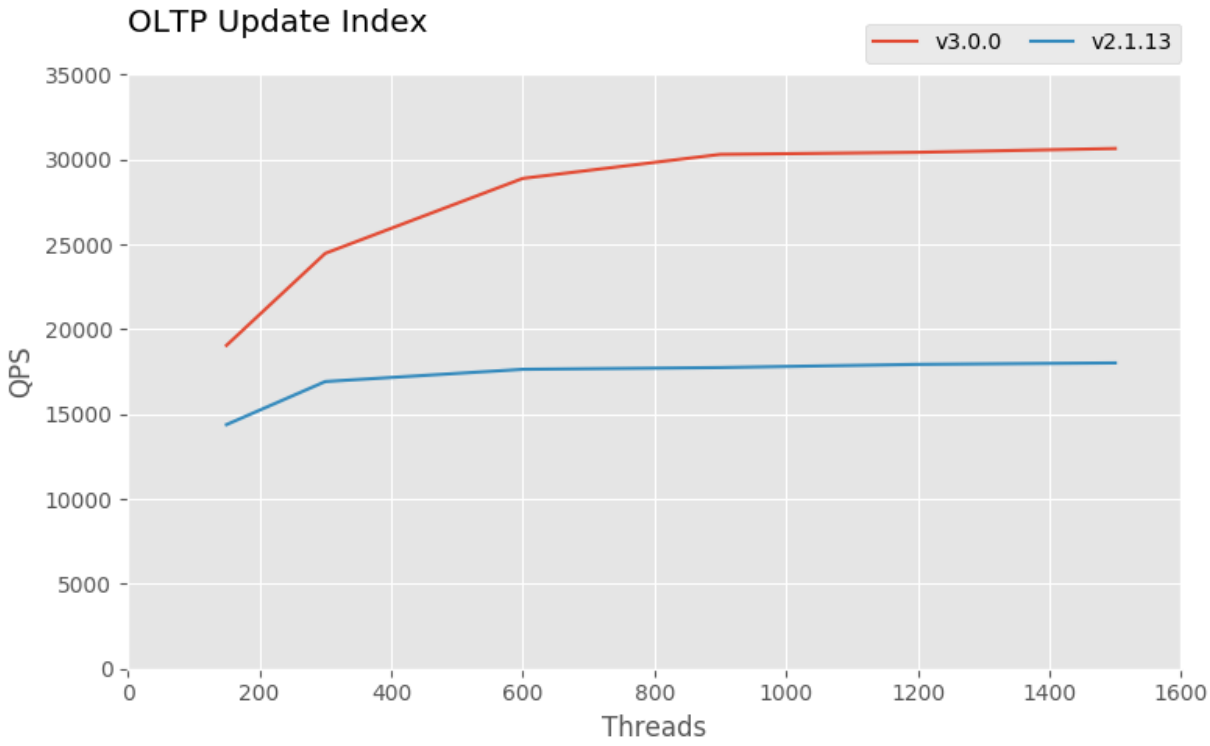

Figure 6: update index

1.2.3.5.4 `Read Write test`

v2.1:

Threads	QPS	95% latency(ms)
150	85140.60	44.98
300	96773.01	82.96
600	105139.81	153.02
900	110041.83	215.44

Threads	QPS	95% latency(ms)
1200	113242.70	277.21
1500	114542.19	337.94

v3.0:

Threads	QPS	95% latency(ms)
150	105692.08	35.59
300	129769.69	58.92
600	141430.86	114.72
900	144371.76	170.48
1200	143344.37	223.34
1500	144567.91	277.21

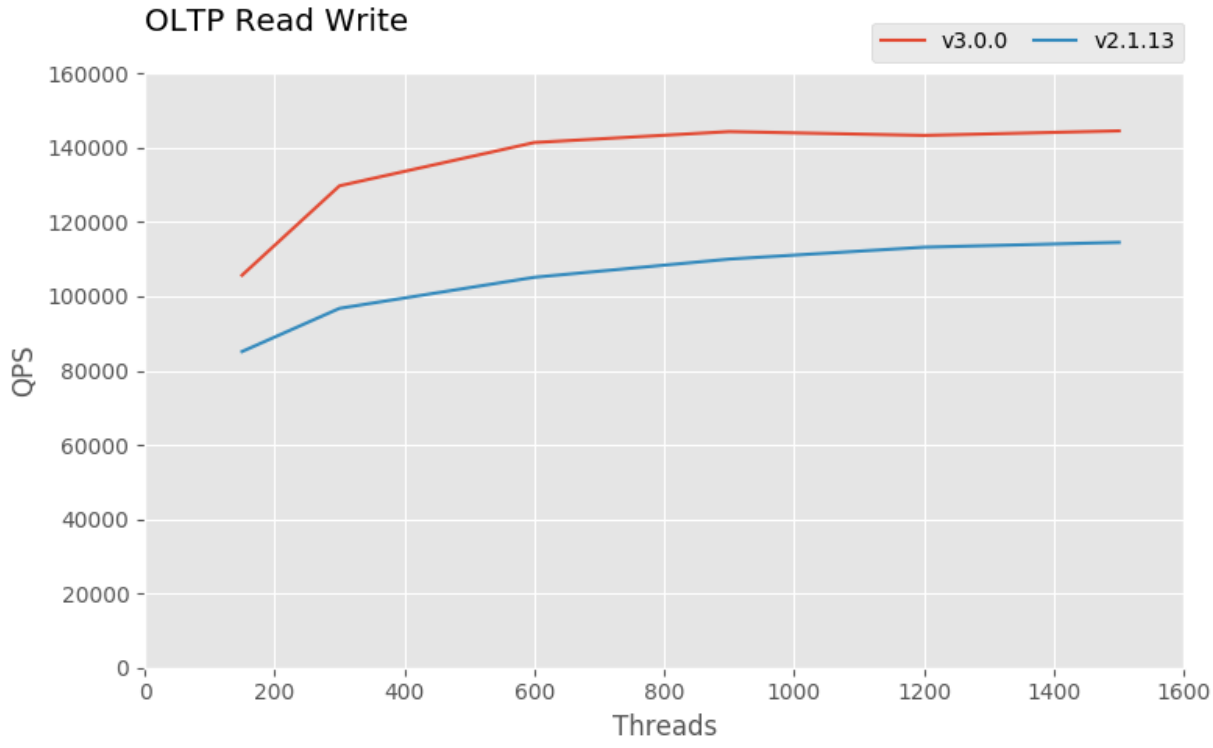

Figure 7: read write

1.2.4 TiDB TPC-C Performance Test Report – v3.0 vs. v2.1

1.2.4.1 Test purpose

This test aims to compare the TPC-C performance of TiDB 3.0 and TiDB 2.1.

1.2.4.2 Test version, time, and place

TiDB version: v3.0.0 vs. v2.1.13

Time: June, 2019

Place: Beijing

1.2.4.3 Test environment

IDC machine:

Type	Name
OS	Linux (CentOS 7.3.1611)
CPU	40 vCPUs, Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
RAM	128GB
DISK	1.5TB SSD * 2

This test uses the open-source BenchmarkSQL 5.0 as the TPC-C testing tool and adds the support for the MySQL protocol. You can download the testing program by using the following command:

```
git clone -b 5.0-mysql-support-opt https://github.com/pingcap/benchmarksql.
    ↪ git
```

1.2.4.4 Test plan

Use BenchmarkSQL to load the data of **1000 warehouses** into the TiDB cluster. By using HAProxy, send concurrent requests to the cluster at an incremental number. A single concurrent test lasts 10 minutes.

1.2.4.4.1 TiDB version information

1.2.4.4.2 v3.0.0

Component	GitHash
TiDB	46c38e15eba43346fb3001280c5034385171ee20
TiKV	a467f410d235fa9c5b3c355e3b620f81d3ac0e0c
PD	70aaa5eee830e21068f1ba2d4c9bae59153e5ca3

1.2.4.4.3 v2.1.13

Component	GitHash
TiDB	6b5b1a6802f9b8f5a22d8aab24ac80729331e1bc

Component	GitHash
TiKV	b3cf3c8d642534ea6fa93d475a46da285cc6acbf
PD	886362ebfb26ef0834935afc57bcee8a39c88e54

1.2.4.4.4 TiDB parameter configuration

```
[log]
level = "error"
[performance]
max-procs = 20
[prepared_plan_cache]
enabled = true
```

1.2.4.4.5 TiKV parameter configuration

The default TiKV configuration is used in both v2.1 and v3.0.

1.2.4.4.6 Cluster topology

Machine IP	Deployment instance
172.16.4.75	2*TiDB 2*TiKV 1*pd
172.16.4.76	2*TiDB 2*TiKV 1*pd
172.16.4.77	2*TiDB 2*TiKV 1*pd

1.2.4.5 Test result

Version	Threads	tpmC
v3.0	128	44068.55
v3.0	256	47094.06
v3.0	512	48808.65
v2.1	128	10641.71
v2.1	256	10861.62
v2.1	512	10965.39

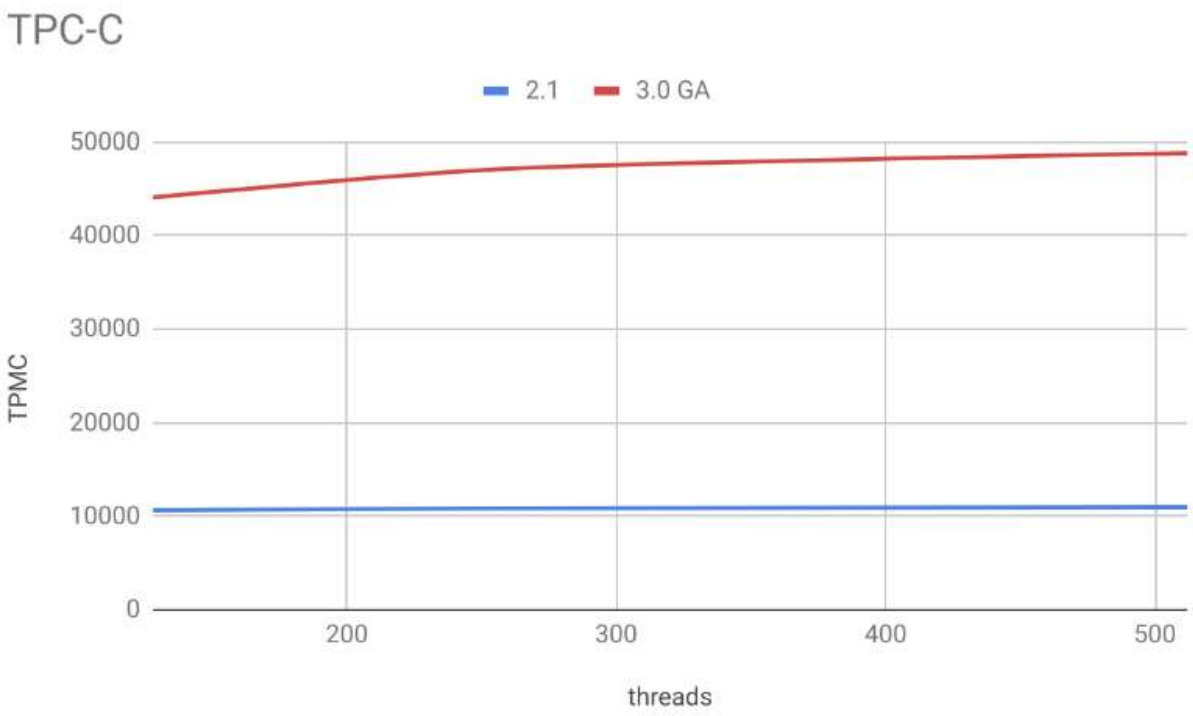

Figure 8: point select

According to the testing statistics, the performance of TiDB 3.0 **has increased by 450%** than that of TiDB 2.1.

1.2.5 Interaction Test on Online Workloads and `ADD INDEX` Operations

1.2.5.1 Test purpose

This document tests the interaction effects between online workloads and `ADD INDEX` operations in the OLTP scenario.

1.2.5.2 Test version, time, and place

TiDB version: v3.0.1

Time: July, 2019

Place: Beijing

1.2.5.3 Test environment

This test runs in a Kubernetes cluster deployed with 3 TiDB instances, 3 TiKV instances and 3 PD instances.

1.2.5.3.1 Version information

Component	GitHash
TiDB	`9e4e8da3c58c65123db5f26409759fe1847529f8`
TiKV	`4151dc8878985df191b47851d67ca21365396133`
PD	`811ce0b9a1335d1b2a049fd97ef9e186f1c9efc1`

Sysbench version: 1.0.17

1.2.5.3.2 TiDB parameter configuration

TiDB, TiKV and PD all use the default TiDB Operator configuration.

1.2.5.3.3 Cluster topology

Machine IP	Deployment instance
172.31.8.8	Sysbench
172.31.7.69, 172.31.5.152, 172.31.11.133	PD
172.31.4.172, 172.31.1.155, 172.31.9.210	TiKV
172.31.7.80, 172.31.5.163, 172.31.11.123	TiDB

1.2.5.3.4 Online workloads simulation using Sysbench

Use Sysbench to import **a table with 2,000,000 rows of data** into the Kubernetes cluster.

Execute the following command to import data:

```
sysbench oltp_common \
    --threads=16 \
    --rand-type=uniform \
    --db-driver=mysql \
    --mysql-db=sbtest \
    --mysql-host=$tidb_host \
    --mysql-port=$tidb_port \
    --mysql-user=root \
    prepare --tables=1 --table-size=2000000
```

Execute the following command to run the test:

```
sysbench $testname \
    --threads=$threads \
    --time=300000 \
    --report-interval=15 \
    --rand-type=uniform \
```

```
    --rand-seed=$RANDOM \
    --db-driver=mysql \
    --mysql-db=sbtest \
    --mysql-host=$tidb_host \
    --mysql-port=$tidb_port \
    --mysql-user=root \
    run --tables=1 --table-size=2000000
```

1.2.5.4 Test plan 1: Frequently perform write operations to the target column of the `ADD INDEX` statement

1. Start the `oltp_read_write` test.
2. Perform at the same time with step 1: use `alter table sbtest1 add index c_idx (c)` to add an index.
3. Perform at the end of step 2: when the index is added successfully, stop the `oltp_read_write` test.
4. Get the duration of `alter table ... add index` and the average TPS and QPS of Sysbench in this period.
5. Gradually increase the value of two parameters `tidb_ddl_reorg_worker_cnt` and `tidb_ddl_reorg_batch_size`, and then repeat step 1-4.

1.2.5.4.1 Test results

Test result of `oltp_read_write` without `ADD INDEX` operations

sysbench TPS	sysbench QPS
350.31	6806

`tidb_ddl_reorg_batch_size = 32`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	402	338.4	6776
2	266	330.3	6001
4	174	288.5	5769
8	129	280.6	5612
16	90	263.5	5273
32	54	229.2	4583
48	57	230.1	4601

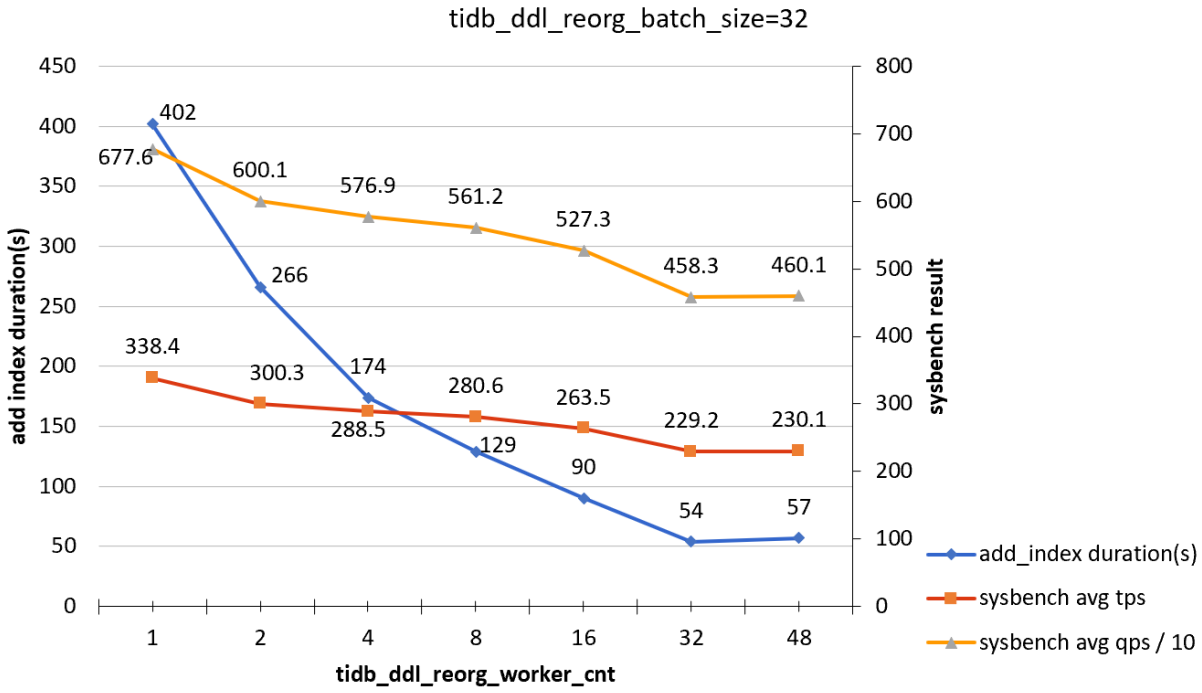

Figure 9: add-index-load-1-b32

`tidb_ddl_reorg_batch_size = 64`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	264	269.4	5388
2	163	266.2	5324
4	105	272.5	5430
8	78	262.5	5228
16	57	215.5	4308
32	42	185.2	3715
48	45	189.2	3794

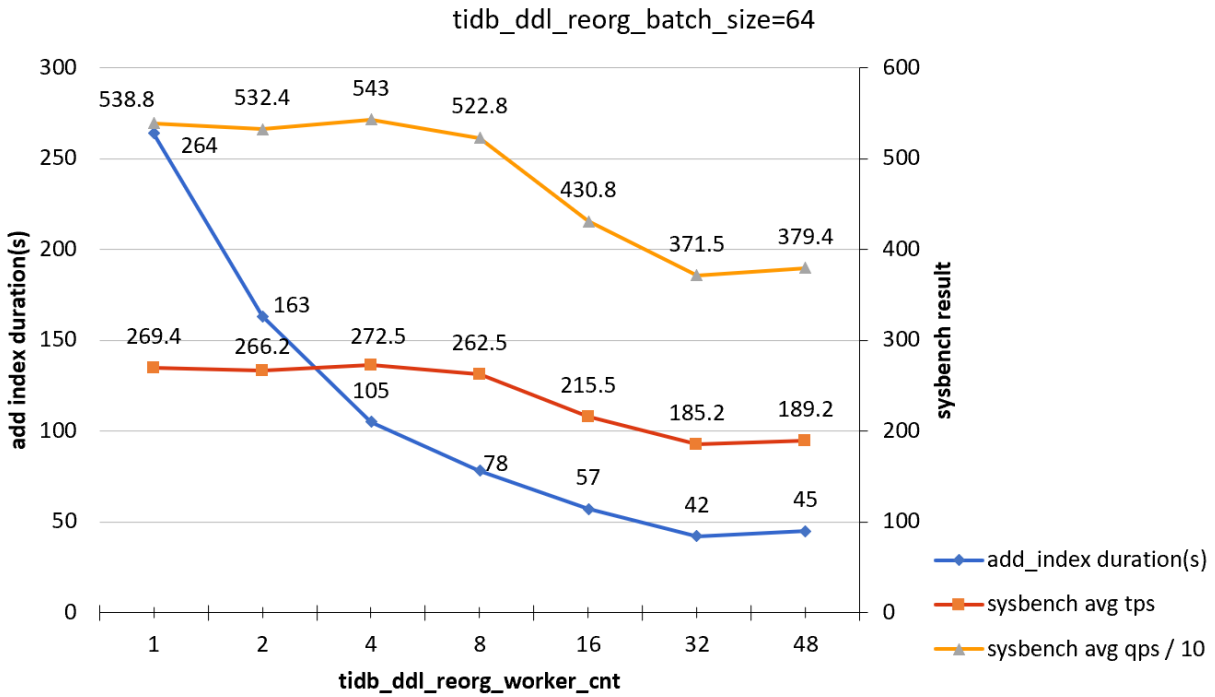

Figure 10: add-index-load-1-b64

`tidb_ddl_reorg_batch_size = 128`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	171	289.1	5779
2	110	274.2	5485
4	79	250.6	5011
8	51	246.1	4922
16	39	171.1	3431
32	35	130.8	2629
48	35	120.5	2425

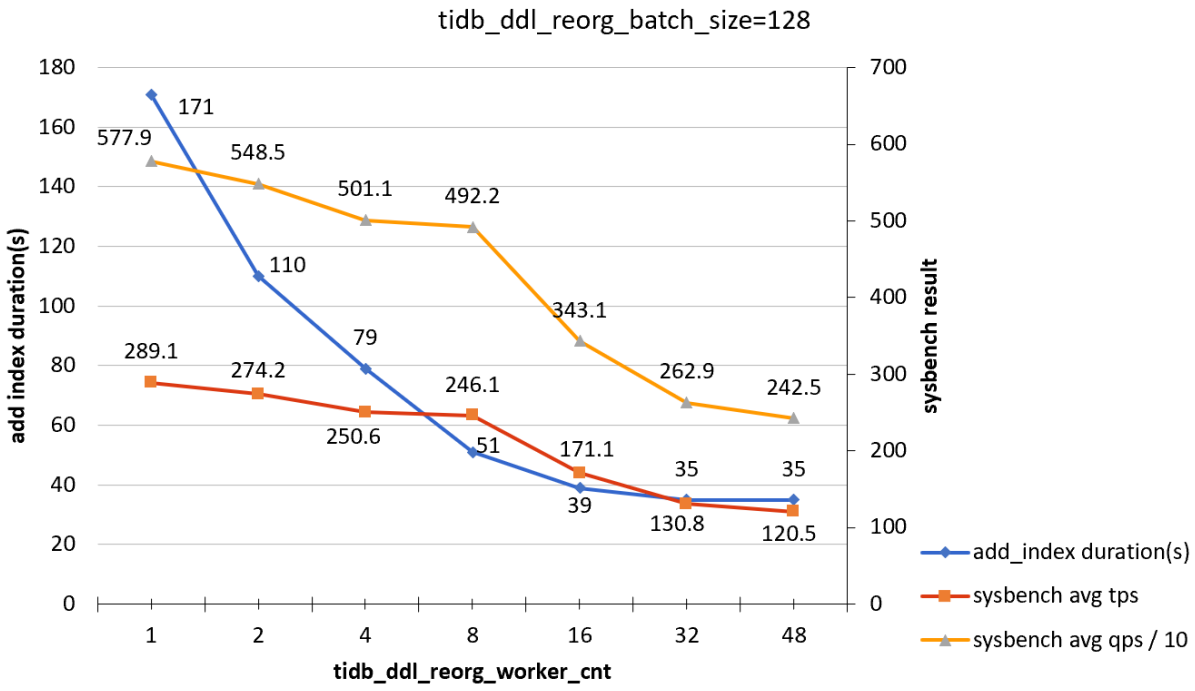

Figure 11: add-index-load-1-b128

`tidb_ddl_reorg_batch_size = 256`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	145	283.0	5659
2	96	282.2	5593
4	56	236.5	4735
8	45	194.2	3882
16	39	149.3	2893
32	36	113.5	2268
48	33	86.2	1715

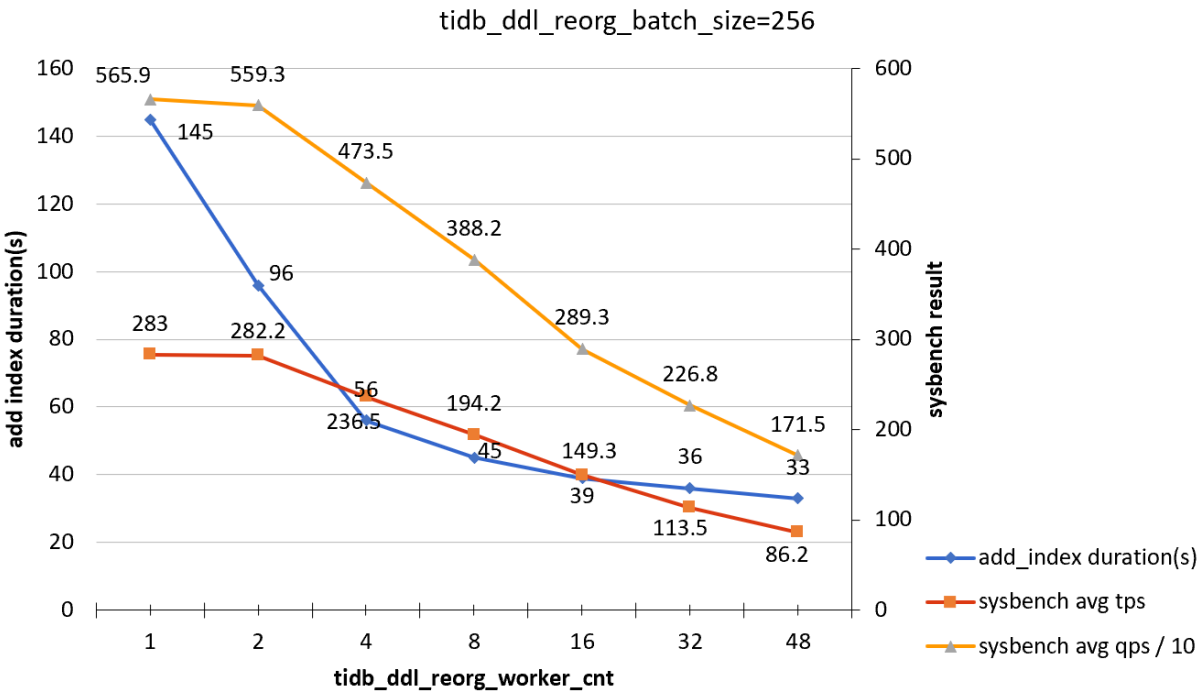

Figure 12: add-index-load-1-b256

`tidb_ddl_reorg_batch_size = 512`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	135	257.8	5147
2	78	252.8	5053
4	49	222.7	4478
8	36	145.4	2904
16	33	109	2190
32	33	72.5	1503
48	33	54.2	1318

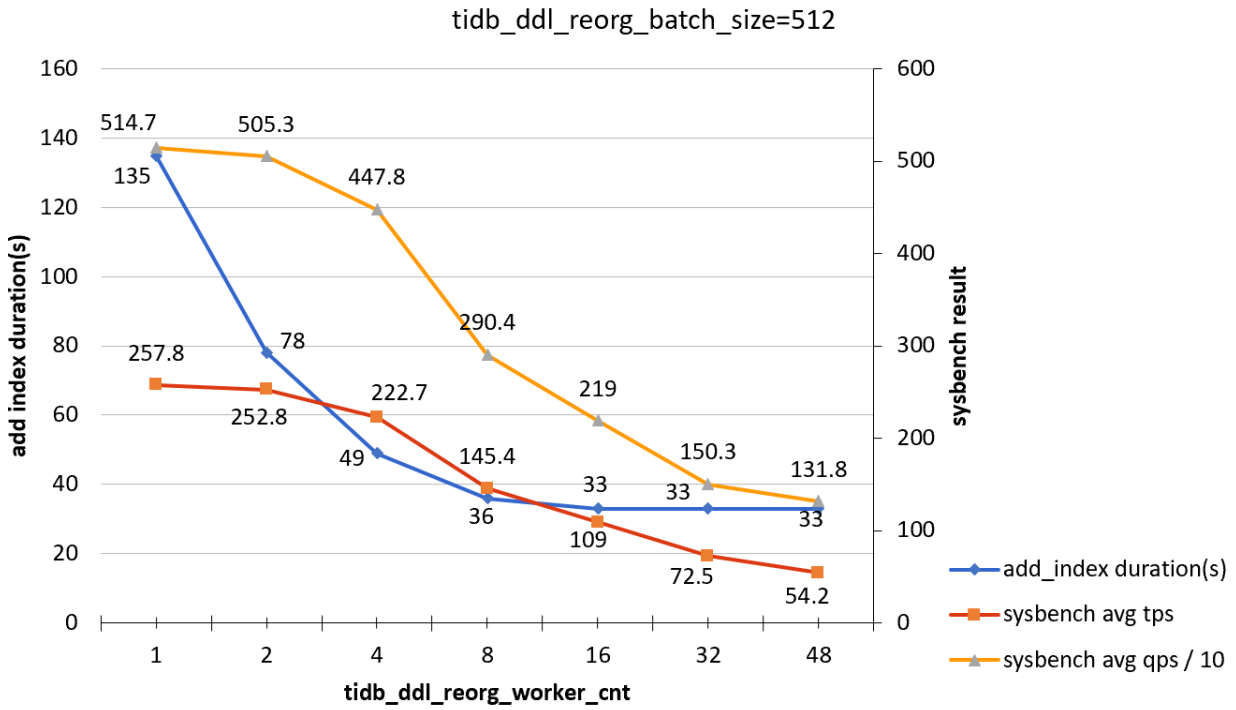

Figure 13: add-index-load-1-b512

`tidb_ddl_reorg_batch_size = 1024`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	111	244.3	4885
2	78	228.4	4573
4	54	168.8	3320
8	39	123.8	2475
16	36	59.6	1213
32	42	93.2	1835
48	51	115.7	2261

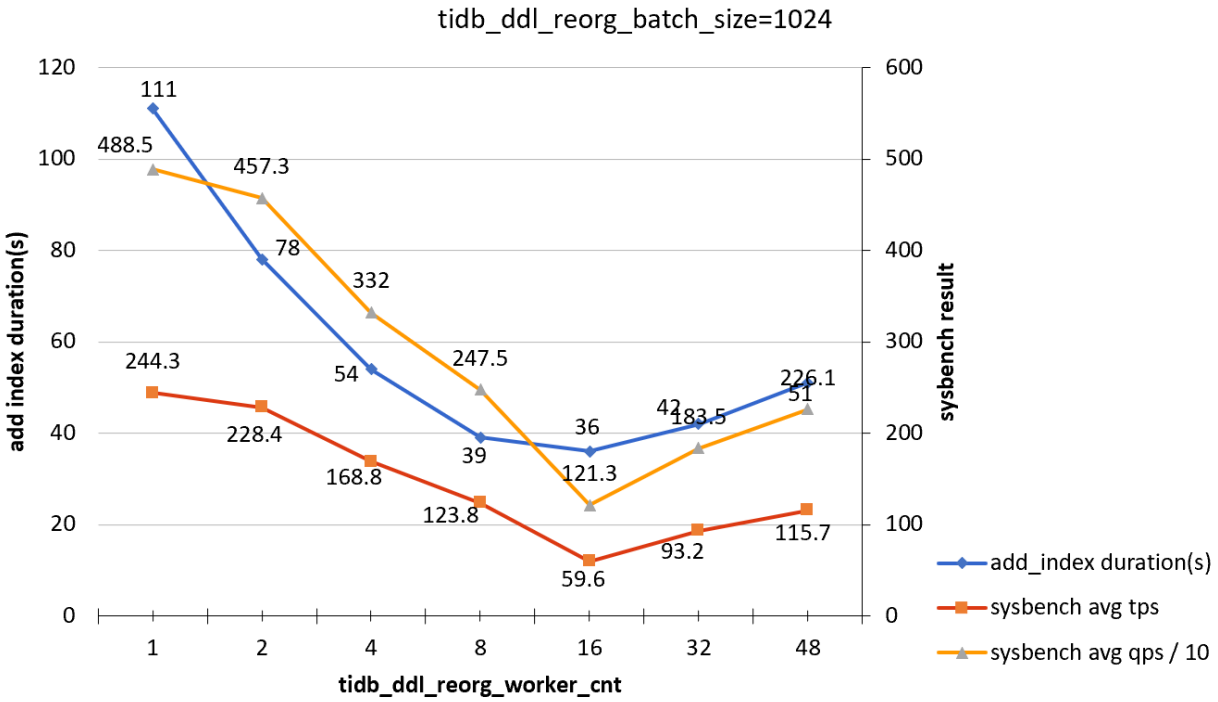

Figure 14: add-index-load-1-b1024

`tidb_ddl_reorg_batch_size = 2048`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	918	243.3	4855
2	1160	209.9	4194
4	342	185.4	3707
8	1316	151.0	3027
16	795	30.5	679
32	1130	26.69	547
48	893	27.5	552

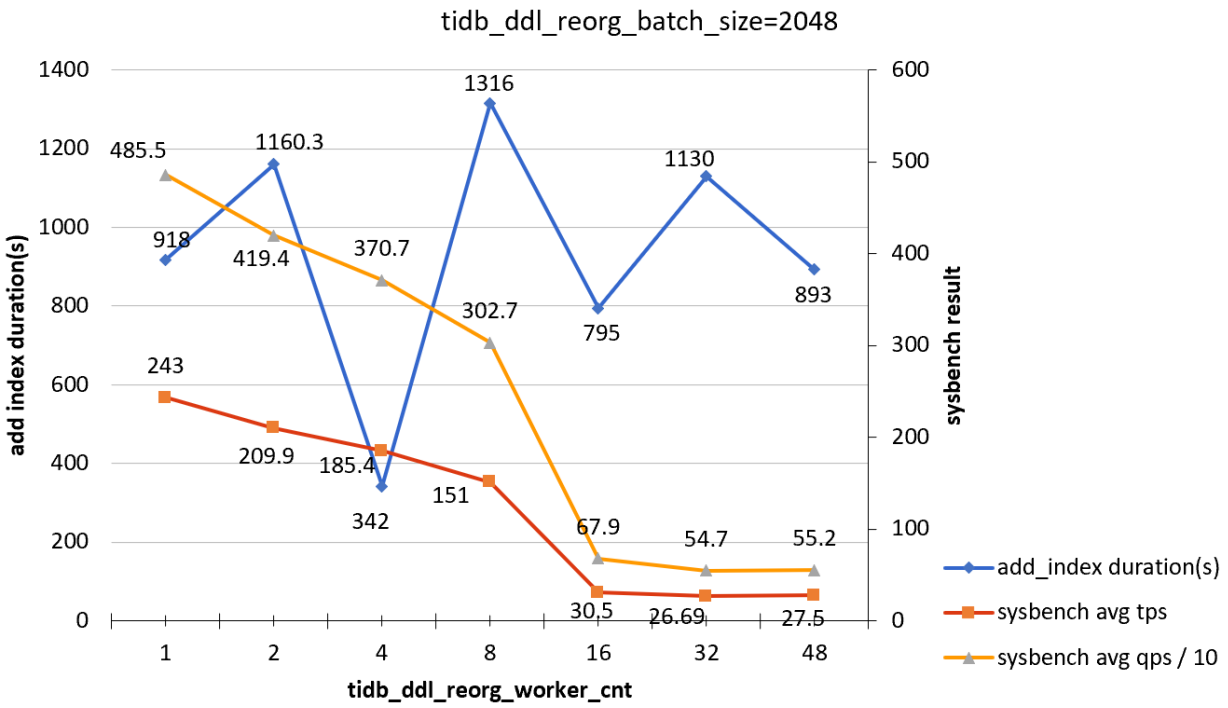

Figure 15: add-index-load-1-b2048

`tidb_ddl_reorg_batch_size = 4096`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	3042	200.0	4001
2	3022	203.8	4076
4	858	195.5	3971
8	3015	177.1	3522
16	837	143.8	2875
32	942	114	2267
48	187	54.2	1416

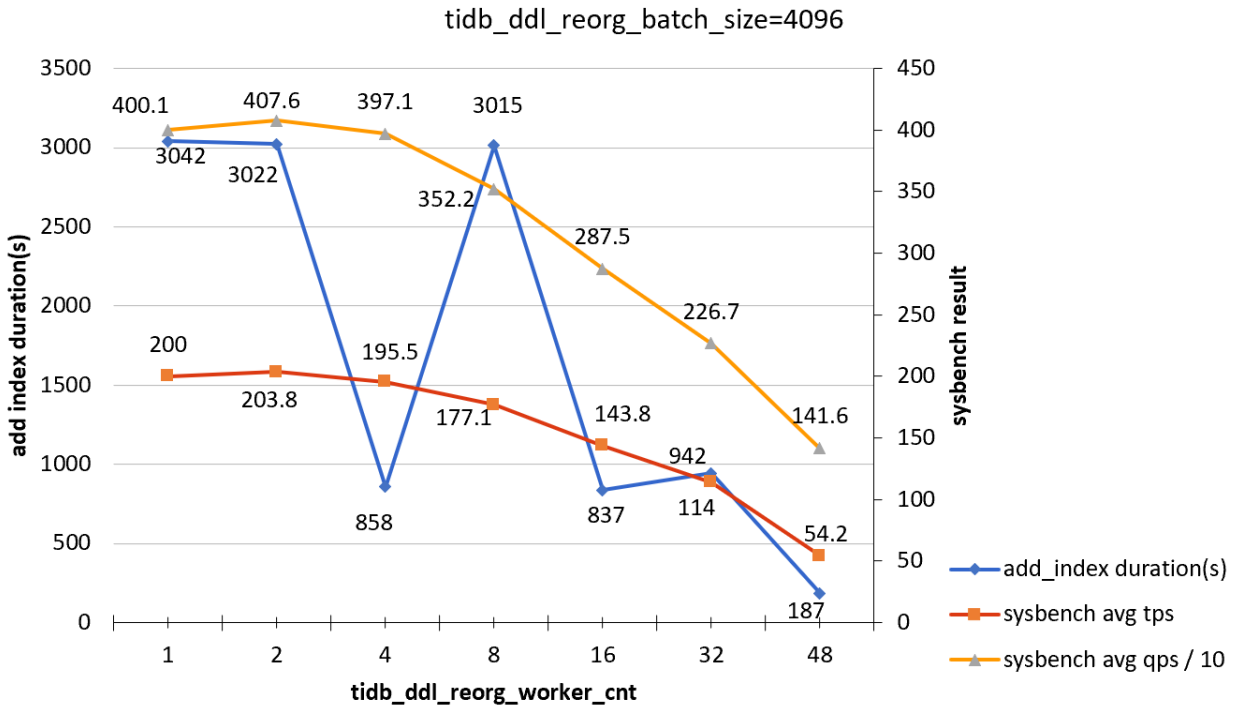

Figure 16: add-index-load-1-b4096

1.2.5.4.2 Test conclusion

When you perform frequent write operations (this test involves `UPDATE`, `INSERT` and `DELETE` operations) to the target column of the `ADD INDEX` statement, the default `ADD INDEX` configuration has a significant impact on the online workload of the system. It is mainly because of the write conflicts caused by the concurrent `ADD INDEX` operation and column update. The performance of the system is as follows:

- As the value of `tidb_ddl_reorg_worker_cnt` and `tidb_ddl_reorg_batch_size` parameters increase, the value of `TiKV_prewrite_latch_wait_duration` increases significantly, slowing down the write speed.
- When the value of `tidb_ddl_reorg_worker_cnt` and `tidb_ddl_reorg_batch_size` are very large, you can execute the `admin show ddl` command to see multiple retry attempts of the DDL job, such as `Write conflict, txnStartTS 410327455965380624 is stale [try again later], ErrCount:38, SnapshotVersion: 410327228136030220`. In this situation, the `ADD INDEX` operation takes a very long time to complete.

1.2.5.5 Test plan 2: Do not perform write operations to the target column of the `ADD INDEX` statement (query-only)

1. Start the `oltp_read_only` test.

2. Perform at the same time with step 1: use `alter table sbtest1 add index c_idx (c)` to add an index.
3. Perform at the end of step 2: when the index is added successfully, stop the `oltp_read_only` test.
4. Get the duration of `alter table ... add index` and the average TPS and QPS of Sysbench in this period.
5. Gradually increase the value of two parameters `tidb_ddl_reorg_worker_cnt` and `tidb_ddl_reorg_batch_size`, and then repeat step 1-4.

1.2.5.5.1 Test results

Test result of `oltp_read_only` without `ADD INDEX` operations

sysbench TPS	sysbench QPS
550.9	8812.8

`tidb_ddl_reorg_batch_size = 32`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	376	548.9	8780
2	212	541.5	8523
4	135	538.6	8549
8	114	536.7	8393
16	77	533.9	8292
32	46	533.4	8103
48	46	532.2	8074

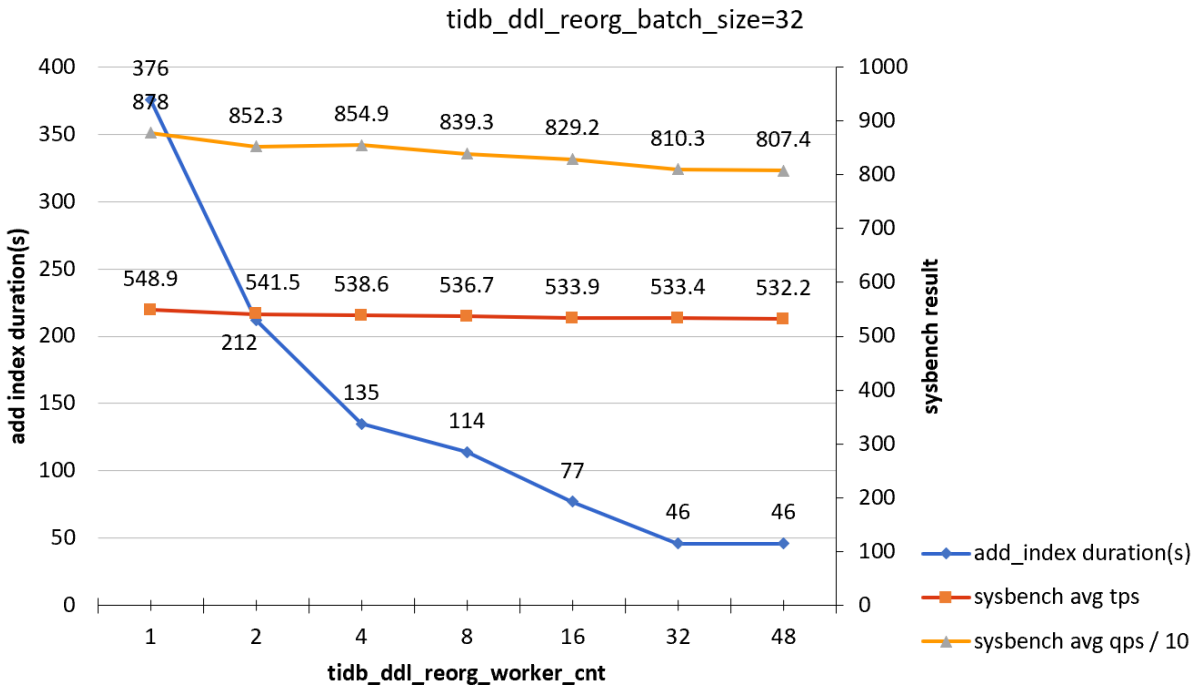

Figure 17: add-index-load-2-b32

`tidb_ddl_reorg_batch_size = 1024`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	91	536.8	8316
2	52	533.9	8165
4	40	522.4	7947
8	36	510	7860
16	33	485.5	7704
32	31	467.5	7516
48	30	562.1	7442

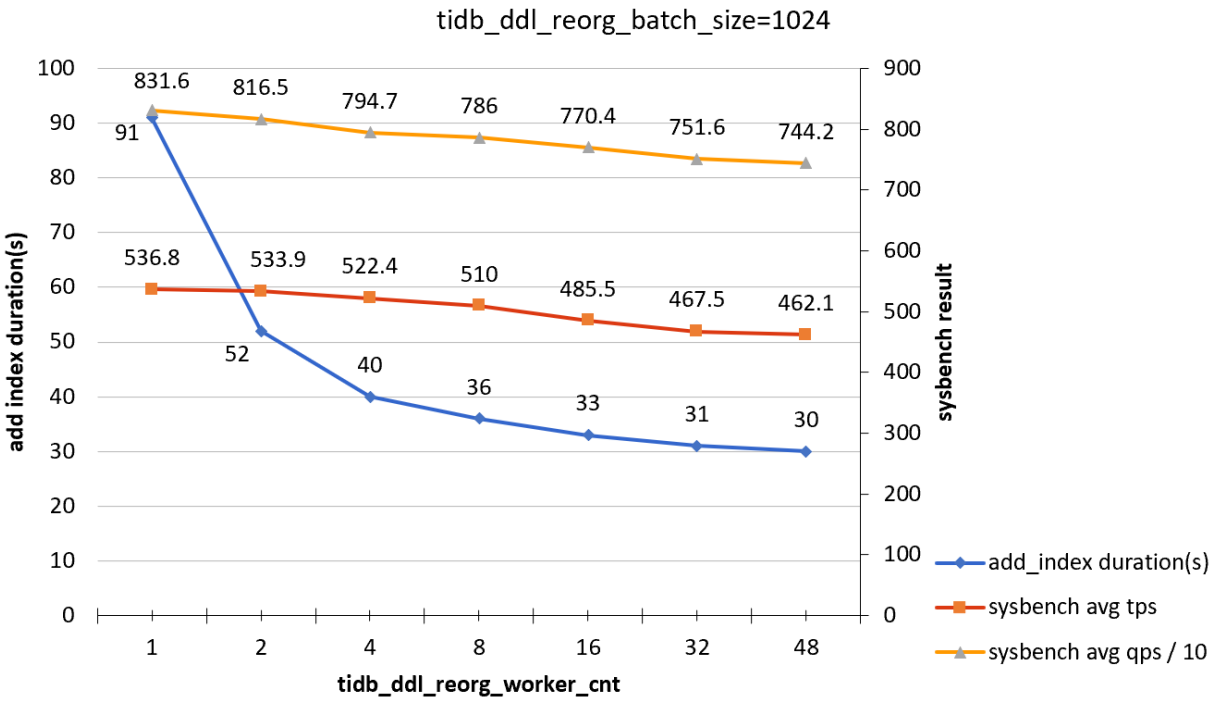

Figure 18: add-index-load-2-b1024

`tidb_ddl_reorg_batch_size = 4096`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	103	502.2	7823
2	63	486.5	7672
4	52	467.4	7516
8	39	452.5	7302
16	35	447.2	7206
32	30	441.9	7057
48	30	440.1	7004

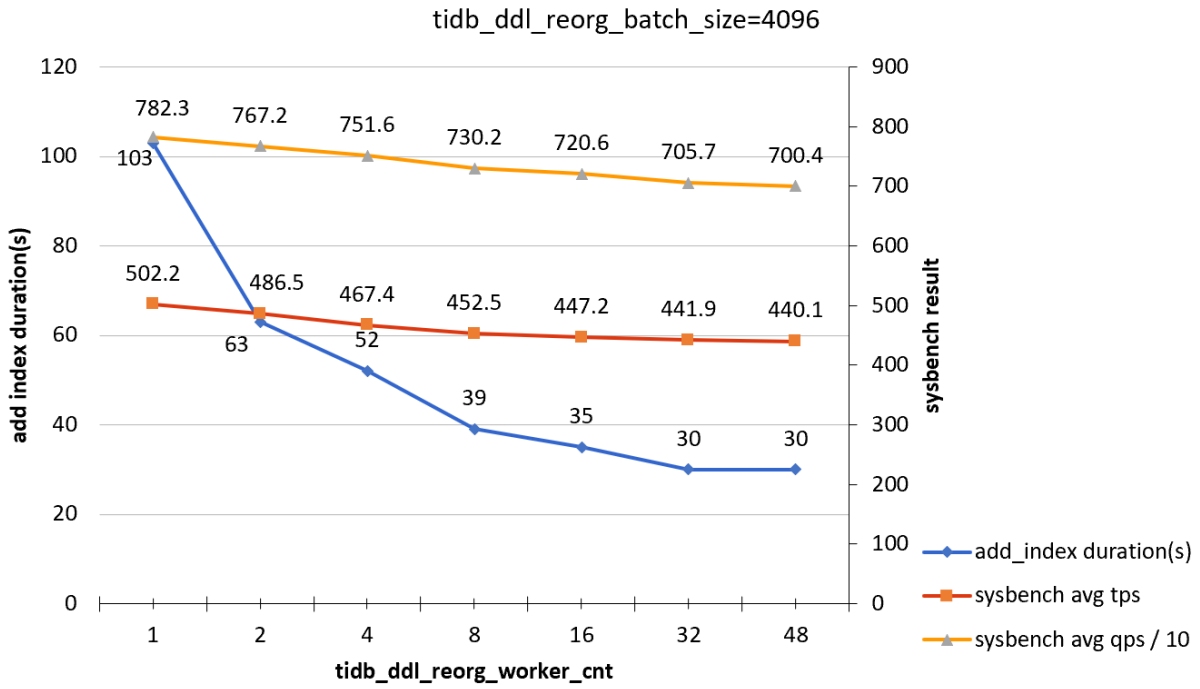

Figure 19: add-index-load-2-b4096

1.2.5.5.2 Test conclusion

When you only perform query operations to the target column of the `ADD INDEX` statement, the effect of `ADD INDEX` operations on online workloads is not obvious.

1.2.5.6 Test plan 3: The target column of the `ADD INDEX` statement is irrelevant to online workloads

1. Start the `oltp_read_write` test.
2. Perform at the same time with step 1: use `alter table test add index pad_idx(pad)` to add an index.
3. Perform at the end of step 2: when the index is added successfully, stop the `oltp_read_only` test.
4. Get the duration of `alter table ... add index` and the average TPS and QPS of Sysbench in this period.
5. Gradually increase the value of two parameters `tidb_ddl_reorg_worker_cnt` and `tidb_ddl_reorg_batch_size`, and then repeat step 1-4.

1.2.5.6.1 Test results

1.2.5.6.2 Test result of `oltp_read_write` without `ADD INDEX` operations

sysbench TPS	sysbench QPS
350.31	6806

```
tidb_ddl_reorg_batch_size = 32
```

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	372	350.4	6892
2	207	344.2	6700
4	140	343.1	6672
8	121	339.1	6579
16	76	340	6607
32	42	343.1	6695
48	42	333.4	6454

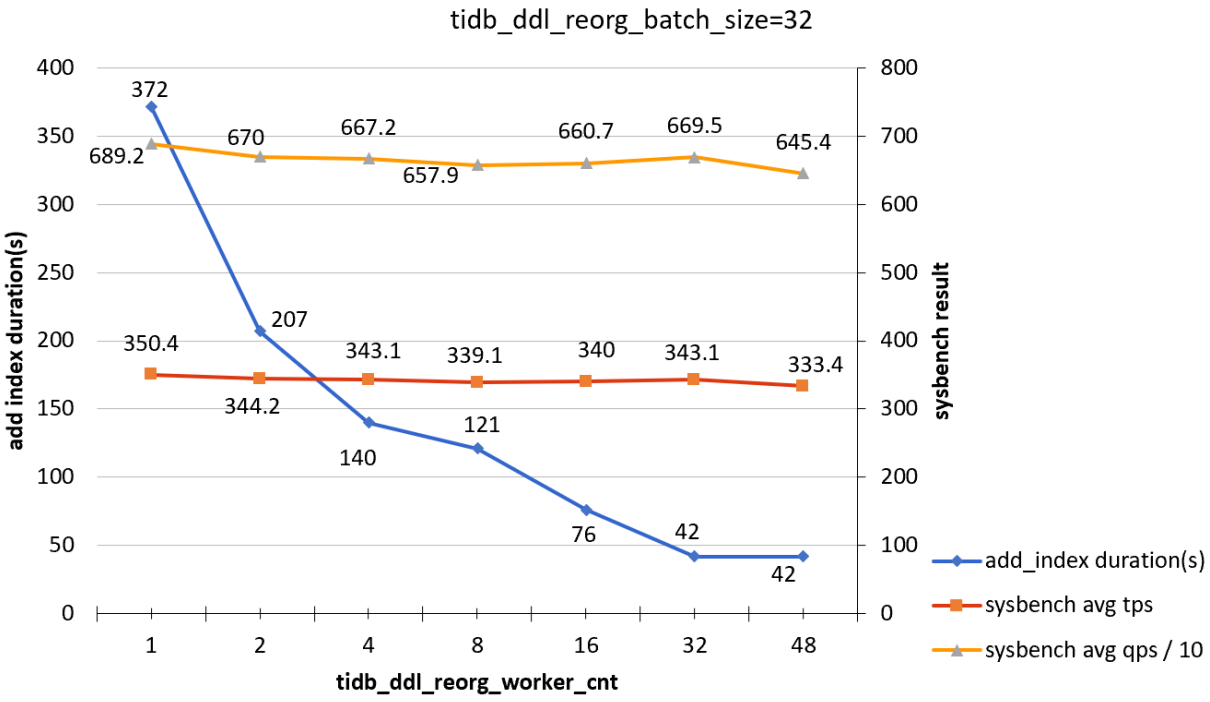

Figure 20: add-index-load-3-b32

```
tidb_ddl_reorg_batch_size = 1024
```

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	94	352.4	6794
2	50	332	6493

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
4	45	330	6456
8	36	325.5	6324
16	32	312.5	6294
32	32	300.6	6017
48	31	279.5	5612

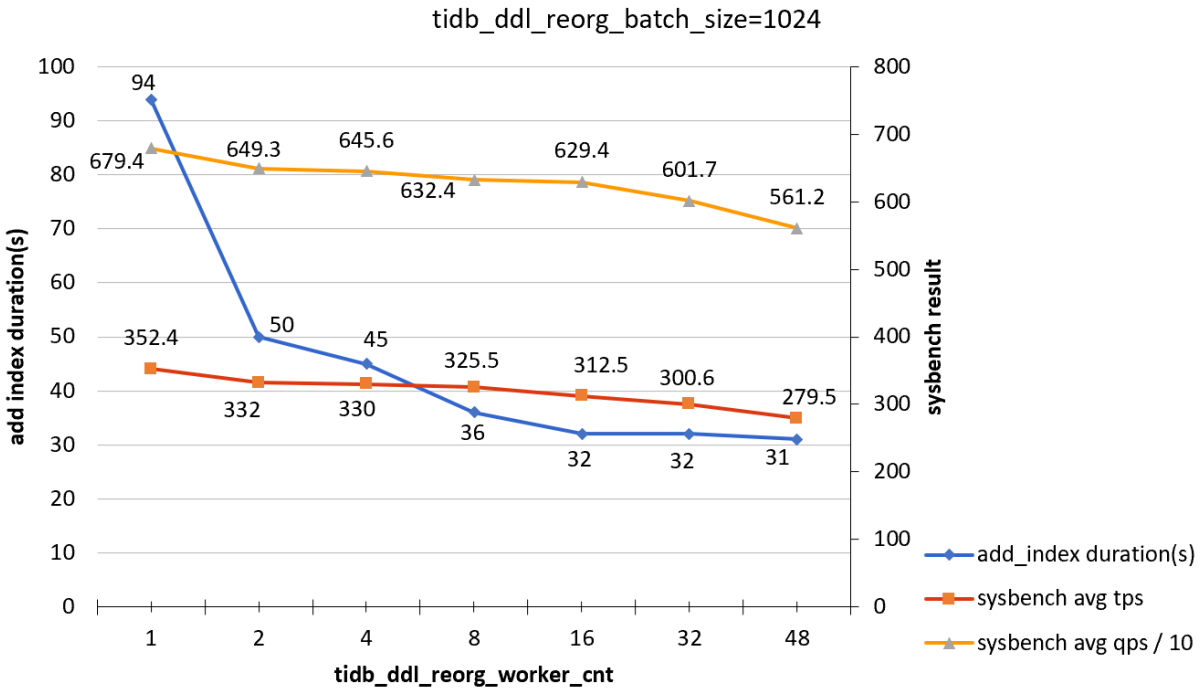

Figure 21: add-index-load-3-b1024

`tidb_ddl_reorg_batch_size = 4096`

tidb_ddl_reorg_worker_cnt	add_index_durations(s)	sysbench TPS	sysbench QPS
1	116	325.5	6324
2	65	312.5	6290
4	50	300.6	6017
8	37	279.5	5612
16	34	250.4	5365
32	32	220.2	4924
48	33	214.8	4544

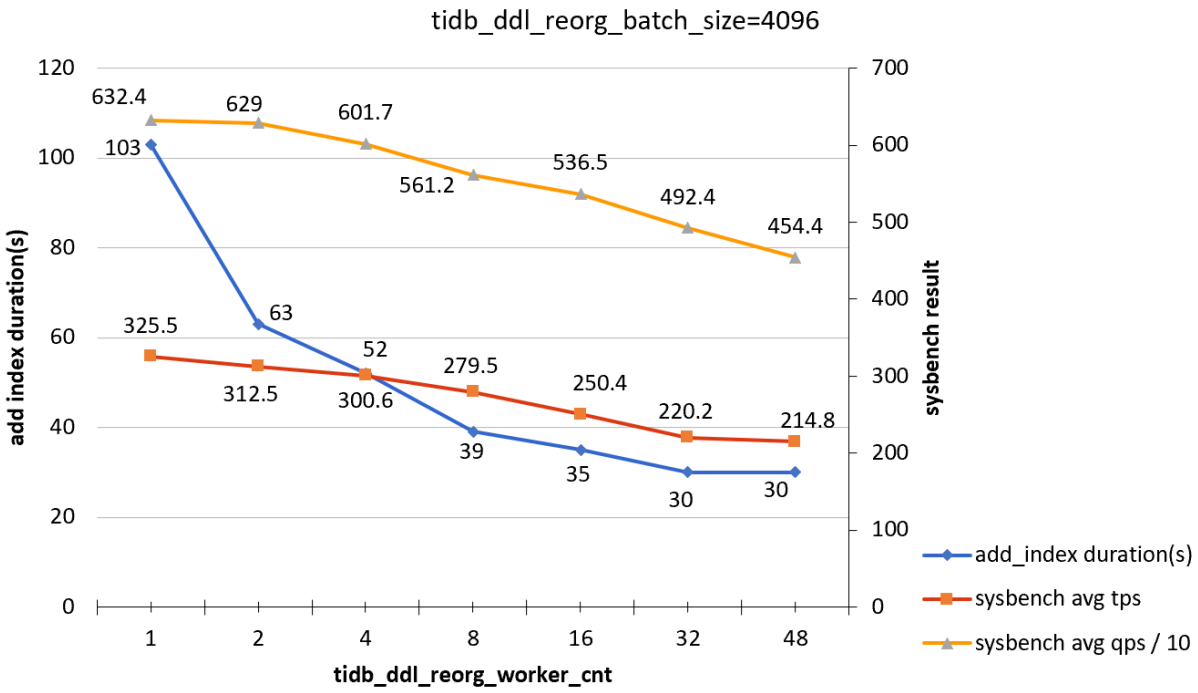

Figure 22: add-index-load-3-b4096

1.2.5.6.3 Test conclusion

When the target column of the `ADD INDEX` statement is irrelevant to online workloads, the effect of `ADD INDEX` operations on the workload is not obvious.

1.2.5.7 Summary

- When you perform frequent write operations (including `INSERT`, `DELETE` and `UPDATE` ↪ operations) to the target column of the `ADD INDEX` statement, the default `ADD` ↪ `INDEX` configuration causes relatively frequent write conflicts, which has a great impact on online workloads. At the same time, the `ADD INDEX` operation takes a long time to complete due to continuous retry attempts. In this test, you can modify the product of `tidb_ddl_reorg_worker_cnt` and `tidb_ddl_reorg_batch_size` to 1/32 of the default value. For example, you can set `tidb_ddl_reorg_worker_cnt` to `4` and `tidb_ddl_reorg_batch_size` to `256` for better performance.
- When only performing query operations to the target column of the `ADD INDEX` statement or the target column is not directly related to online workloads, you can use the default `ADD INDEX` configuration.

2 Concepts

2.1 TiDB Architecture

The TiDB platform is comprised of three key components: the TiDB server, the PD server, and the TiKV server. In addition, TiDB also provides the TiSpark component for complex OLAP requirements and the TiDB Operator to make things simpler for the deployment and management on cloud.

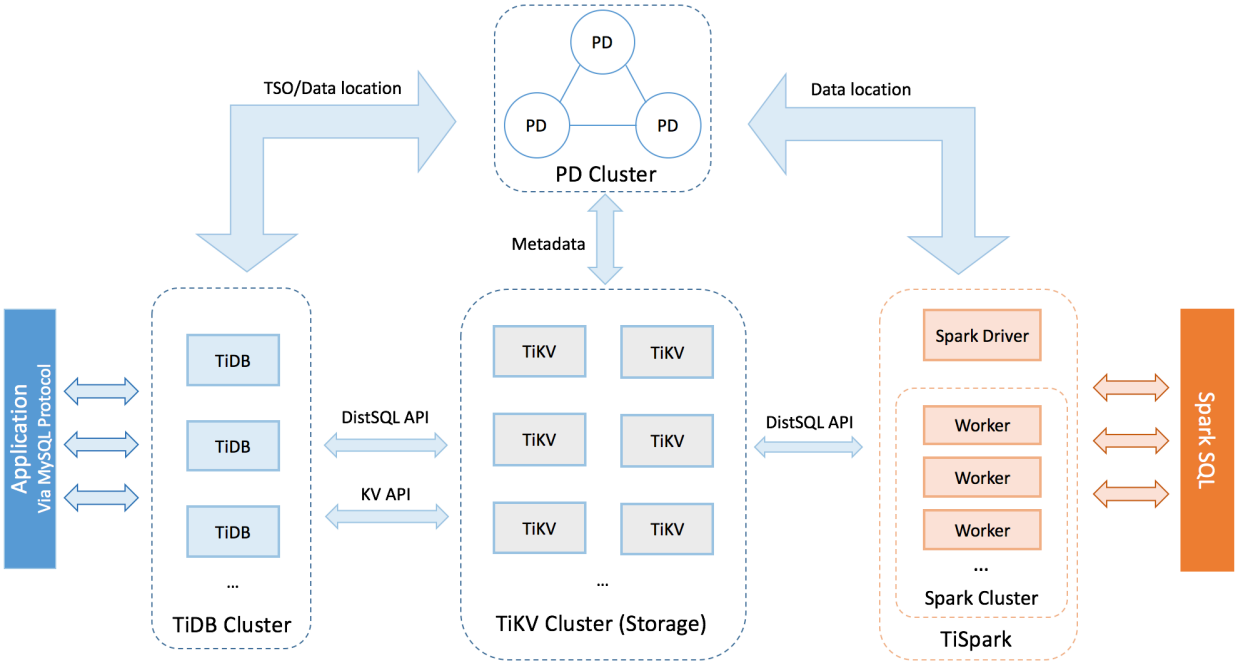

Figure 23: image alt text

2.1.1 TiDB server

The TiDB server is in charge of the following operations:

1. Receiving the SQL requests
2. Processing the SQL related logics
3. Locating the TiKV address for storing and computing data through Placement Driver (PD)
4. Exchanging data with TiKV
5. Returning the result

The TiDB server is stateless. It does not store data and it is for computing only. TiDB is horizontally scalable and provides the unified interface to the outside through the load balancing components such as Linux Virtual Server (LVS), HAProxy, or F5.

2.1.2 Placement Driver server

The Placement Driver (PD) server is the managing component of the entire cluster and is in charge of the following three operations:

1. Storing the metadata of the cluster such as the region location of a specific key.
2. Scheduling and load balancing regions in the TiKV cluster, including but not limited to data migration and Raft group leader transfer.
3. Allocating the transaction ID that is globally unique and monotonically increasing.

The PD server ensures redundancy by using the Raft consensus algorithm. The Raft leader is responsible for handling all operations, with remaining PD servers available for high availability only. It is recommended to deploy PD as an odd number of nodes.

2.1.3 TiKV server

The TiKV server is responsible for storing data. From an external view, TiKV is a distributed transactional Key-Value storage engine. Region is the basic unit to store data. Each Region stores the data for a particular Key Range which is a left-closed and right-open interval from StartKey to EndKey. There are multiple Regions in each TiKV node. TiKV uses the Raft protocol for replication to ensure the data consistency and disaster recovery. The replicas of the same Region on different nodes compose a Raft Group. The load balancing of the data among different TiKV nodes is carried out by PD through scheduling the load in units of Region.

2.1.4 TiSpark

TiSpark deals with the complex OLAP requirements. TiSpark makes Spark SQL directly run on the storage layer of the TiDB cluster, combines the advantages of the distributed TiKV cluster, and integrates into the big data ecosystem. With TiSpark, TiDB can support both OLTP and OLAP scenarios in one cluster, so the users never need to worry about data replication.

2.1.5 TiDB Operator

TiDB Operator empowers TiDB users to deploy and manage TiDB clusters on mainstream cloud infrastructure (Kubernetes).

TiDB Operator:

- Combines the best practices of the container orchestration from the cloud-native community with the know-how of TiDB operation and maintenance.
- Capable of quick deployment, mixed deployment among multiple clusters, automatic operation and maintenance, automatic fail-over, etc.
- Makes it user-friendly to use and manage TiDB.

2.2 Key Features

2.2.1 Key Features

2.2.1.1 Horizontal scalability

TiDB expands both SQL processing and storage by simply adding new nodes. This makes infrastructure capacity planning both easier and more cost-effective than traditional relational databases which only scale vertically.

2.2.1.2 MySQL compatible syntax

TiDB acts like it is a MySQL 5.7 server to your applications. You can continue to use all of the existing MySQL client libraries, and in many cases, you will not need to change a single line of code in your application.

TiDB does not have 100% MySQL compatibility because we built the layer from scratch in order to maximize the performance advantages inherent to a distributed system. We believe in being transparent about the level of MySQL compatibility that TiDB provides. Please check out the list of known compatibility differences.

2.2.1.3 Replicate from and to MySQL

TiDB supports the ability to replicate from a MySQL or MariaDB installation, using its Data Migration (DM) toolchain. Replication is also possible in the direction of TiDB to MySQL using the TiDB Binlog.

We believe that being able to replicate in both directions lowers the risk when either evaluating or migrating to TiDB from MySQL.

2.2.1.4 Distributed transactions with strong consistency

TiDB internally shards table into small range-based chunks that we refer to as "Regions". Each Region defaults to approximately 100MiB in size, and TiDB uses a Two-phase commit internally to ensure that Regions are maintained in a transactionally consistent way.

Transactions in TiDB are strongly consistent, with snapshot isolation level consistency. For more information, see transaction behavior and performance differences. This makes TiDB more comparable to traditional relational databases in semantics than some of the newer NoSQL systems using eventual consistency.

These behaviors are transparent to your application(s), which only need to connect to TiDB using a MySQL 5.7 compatible client library.

2.2.1.5 Cloud native architecture

TiDB is designed to work in the cloud – public, private, or hybrid – making deployment, provisioning, operations, and maintenance simple.

The storage layer of TiDB, called TiKV, became a Cloud Native Computing Foundation member project in 2018. The architecture of the TiDB platform also allows SQL processing and storage to be scaled independently of each other in a very cloud-friendly manner.

2.2.1.6 Minimize ETL with HTAP

TiDB is designed to support both transaction processing (OLTP) and analytical processing (OLAP) workloads. This means that while you may have traditionally transacted on MySQL and then Extracted, Transformed and Loaded (ETL) data into a column store for analytical processing, this step is no longer required.

With trends in business such as moving from two-day delivery to instant, it is important to be able to run analytics with minimal delay. The future is in HTAP databases which can perform the *hybrid* of Transactional and Analytical processing.

2.2.1.7 Fault tolerance & recovery with Raft

TiDB uses the Raft consensus algorithm to ensure that data is safely replicated throughout storage in Raft groups. In the event of failure, a Raft group will automatically elect a new leader for the failed member, and self-heal the TiDB cluster without any required manual intervention.

Failure and self-healing operations are also transparent to applications. TiDB servers will retry accessing the data after the leadership change, with the only impact being slightly higher latency for queries attempting to access this specific data in between when the failure is detected and fixed.

2.2.1.8 Automatic rebalancing

The storage in TiKV is automatically rebalanced to match changes in your workload. For example, if part of your data is more frequently accessed, this hotspot will be detected and may trigger the data to be rebalanced among other TiKV servers. Chunks of data ("Regions" in TiDB terminology) will automatically be split or merged as needed.

This helps remove some of the headaches associated with maintaining a large database cluster and also leads to better utilization over traditional source-replica read-write splitting that is commonly used with MySQL deployments.

2.2.1.9 Deployment and orchestration with Ansible, Kubernetes, Docker

TiDB supports several deployment and orchestration methods, like Ansible, Kubernetes, and Docker. Whether your environment is bare metal, virtualized or containerized, TiDB can be deployed, upgraded, operated, and maintained using the best toolset most suited to your needs.

2.2.1.10 JSON support

TiDB supports a built-in JSON data type and set of built-in functions to search, manipulate and create JSON data. This enables you to build your application without enforcing a strict schema up front.

2.2.1.11 Spark integration

TiDB natively supports an Apache Spark plug-in, called TiSpark, with a SparkSQL interface that enables users to run analytical workloads using Spark directly on TiKV, where the data is stored. This plug-in does not interfere with transactional processing in the TiDB server. This integration takes advantage of TiDB's modular architecture to support HTAP workloads.

2.2.1.12 Read historical data without restoring from backup

Many restore-from-backup events are the result of accidental deletion or modification of the wrong data. With TiDB, you can access the older versions from MVCC by specifying a timestamp in the past from when you would like to access the data.

Your session will be placed in read-only mode while reading the earlier versions of rows, but you can use this to export the data and reload it to the current time if required.

2.2.1.13 Fast import and restore of data

TiDB supports the ability to fast-import both Mydumper and .csv formatted data using an optimized insert mode that disables redo logging, and applies a number of optimizations.

With TiDB Lightning, you can import data into TiDB at over 100GiB/hour using production-grade hardware.

2.2.1.14 Hybrid of column and row storage

TiDB supports the ability to store data in both row-oriented and (coming soon) column-oriented storage. This enables a wide spectrum of both transactional and analytical queries to be executed efficiently in TiDB and TiSpark. The TiDB optimizer is also able to determine which queries are best served by column storage, and route the queries appropriately.

2.2.1.15 SQL plan management

In both MySQL and TiDB, optimizer hints are available to override the default query execution plan with a better known plan. The problem with this approach is that it requires an application developer to make modifications to query text to inject the hint. This can also be undesirable in the case that an ORM is used to generate the query.

In TiDB 3.0, you will be able to bind queries to a specific execution plan directly within the TiDB server. This method is entirely transparent to application code.

2.2.1.16 Open source

TiDB has been released under the Apache 2.0 license since its initial launch in 2015. The TiDB server has (to our knowledge) the highest contributor count on GitHub of any relational database project.

2.2.1.17 Online schema changes

TiDB implements the *Online, Asynchronous Schema Change* algorithm first described in Google's F1 paper.

In simplified terms, this means that TiDB is able to make changes to the schema across its distributed architecture without blocking either read or write operations. There is no need to use an external schema change tool or flip between sources and replicas as is common in large MySQL deployments.

3 How-to

3.1 Get Started

3.1.1 Quick Start with TiDB

To try out the TiDB database, see Quick Start Guide for the TiDB Database Platform to deploy the latest stable version of TiDB. The deployment methods introduced in this guide is only for quick trial, not for the production environment.

3.1.2 Explore SQL with TiDB

TiDB is compatible with MySQL, you can use MySQL statements directly in most of the cases. For unsupported features, see Compatibility with MySQL.

To experiment with SQL and test out TiDB compatibility with MySQL queries, you can run TiDB directly in your web browser without installing it. You can also first deploy a TiDB cluster and then run SQL statements in it.

This page walks you through the basic TiDB SQL statements such as DDL, DML and CRUD operations. For a complete list of TiDB statements, see TiDB SQL Syntax Diagram.

3.1.2.1 Create, show, and drop a database

3.1.2.1.1 Create a database

To create a database, use the `CREATE DATABASE` statement:

```
CREATE DATABASE db_name [options];
```

For example, to create a database named `samp_db`:

```
CREATE DATABASE IF NOT EXISTS samp_db;
```

3.1.2.1.2 Show the databases

To show the databases, use the `SHOW DATABASES` statement:

```
SHOW DATABASES;
```

3.1.2.1.3 Delete a database

To delete a database, use the `DROP DATABASE` statement:

```
DROP DATABASE samp_db;
```

3.1.2.2 Create, show, and drop a table

3.1.2.2.1 Create a table

- To create a table, use the `CREATE TABLE` statement:

  ```
  CREATE TABLE table_name column_name data_type constraint;
  ```

  For example:

  ```
  CREATE TABLE person (
      number INT(11),
      name VARCHAR(255),
      birthday DATE
  );
  ```

- Add `IF NOT EXISTS` to prevent an error if the table exists:

  ```
  CREATE TABLE IF NOT EXISTS person (
      number INT(11),
      name VARCHAR(255),
      birthday DATE
  );
  ```

- To view the statement that creates the table, use the `SHOW CREATE` statement:

```
SHOW CREATE table person;
```

3.1.2.2.2 Show the tables

- To show all the tables in a database, use the `SHOW TABLES` statement:

```
SHOW TABLES FROM samp_db;
```

- To show all the columns in a table, use the `SHOW FULL COLUMNS` statement:

```
SHOW FULL COLUMNS FROM person;
```

3.1.2.2.3 Delete a table

To delete a table, use the `DROP TABLE` statement:

```
DROP TABLE person;
```

or

```
DROP TABLE IF EXISTS person;
```

3.1.2.3 Create, show, and drop an index

3.1.2.3.1 Create an index

- To create an index for the column whose value is not unique, use the `CREATE INDEX` or `ALTER TABLE` statement:

```
CREATE INDEX person_num ON person (number);
```

or

```
ALTER TABLE person ADD INDEX person_num (number);
```

- To create a unique index for the column whose value is unique, use the `CREATE UNIQUE INDEX` or `ALTER TABLE` statement:

```
CREATE UNIQUE INDEX person_num ON person (number);
```

or

```
ALTER TABLE person ADD UNIQUE person_num (number);
```

3.1.2.3.2 Show the indexes

To show all the indexes in a table, use the `SHOW INDEX` statement:

```
SHOW INDEX FROM person;
```

3.1.2.3.3 Delete an index

To delete an index, use the `DROP INDEX` or `ALTER TABLE` statement:

```
DROP INDEX person_num ON person;
```

```
ALTER TABLE person DROP INDEX person_num;
```

3.1.2.4 Insert, select, update, and delete data

3.1.2.4.1 Insert data

To insert data into a table, use the `INSERT` statement:

```
INSERT INTO person VALUES("1","tom","20170912");
```

3.1.2.4.2 Select data

To view the data in a table, use the `SELECT` statement:

```
SELECT * FROM person;
```

```
+--------+------+------------+
| number | name | birthday |
+--------+------+------------+
|      1 | tom | 2017-09-12 |
+--------+------+------------+
```

3.1.2.4.3 Update data

To update the data in a table, use the `UPDATE` statement:

```
UPDATE person SET birthday='20171010' WHERE name='tom';
```

```
SELECT * FROM person;
```

```
+--------+------+------------+
| number | name | birthday |
+--------+------+------------+
|      1 | tom | 2017-10-10 |
+--------+------+------------+
```

3.1.2.4.4 Delete data

To delete the data in a table, use the `DELETE` statement:

```
DELETE FROM person WHERE number=1;
```

```
SELECT * FROM person;
```

```
Empty set (0.00 sec)
```

3.1.2.5 Create, authorize, and delete a user

3.1.2.5.1 Create a user

To create a user, use the `CREATE USER` statement. The following example creates a user named `tiuser` with the password `123456`:

```
CREATE USER 'tiuser'@'localhost' IDENTIFIED BY '123456';
```

3.1.2.5.2 Authorize a user

- To grant `tiuser` the privilege to retrieve the tables in the `samp_db` database:

  ```
  GRANT SELECT ON samp_db.* TO 'tiuser'@'localhost';
  ```

- To check the privileges of `tiuser`:

  ```
  SHOW GRANTS for tiuser@localhost;
  ```

3.1.2.5.3 Delete a user

To delete `tiuser`:

```
DROP USER 'tiuser'@'localhost';
```

3.1.3 Import Example Database

Examples used in the TiDB manual use System Data from Capital Bikeshare, released under the Capital Bikeshare Data License Agreement.

3.1.3.1 Download all data files

The system data is available for download in .zip files organized per year. Downloading and extracting all files requires approximately 3GB of disk space. To download all files for years 2010-2017 using a bash script:

```
mkdir -p bikeshare-data &&
cd bikeshare-data &&
curl -L --remote-name-all https://s3.amazonaws.com/capitalbikeshare-data
    ↪ /{2010..2017}-capitalbikeshare-tripdata.zip &&
unzip \*-tripdata.zip
```

3.1.3.2 Load data into TiDB

The system data can be imported into TiDB using the following schema:

```
CREATE DATABASE bikeshare;
USE bikeshare;

CREATE TABLE trips (
 trip_id bigint NOT NULL PRIMARY KEY AUTO_INCREMENT,
 duration integer not null,
 start_date datetime,
 end_date datetime,
 start_station_number integer,
 start_station varchar(255),
 end_station_number integer,
 end_station varchar(255),
 bike_number varchar(255),
 member_type varchar(255)
);
```

You can import files individually using the example `LOAD DATA` command here, or import all files using the bash loop below:

```
LOAD DATA LOCAL INFILE '2017Q1-capitalbikeshare-tripdata.csv' INTO TABLE
    ↪ trips
  FIELDS TERMINATED BY ',' ENCLOSED BY '"'
  LINES TERMINATED BY '\r\n'
  IGNORE 1 LINES
(duration, start_date, end_date, start_station_number, start_station,
end_station_number, end_station, bike_number, member_type);
```

3.1.3.2.1 Import all files

> **Note:**
>
> When you start the MySQL client, use the `--local-infile=1` option.

To import all `*.csv` files into TiDB in a bash loop:

```
for FILE in `ls *.csv`; do
 echo "== $FILE =="
 mysql bikeshare --local-infile=1 -e "LOAD DATA LOCAL INFILE '${FILE}' INTO
    ↪ TABLE trips FIELDS TERMINATED BY ',' ENCLOSED BY '\"' LINES
    ↪ TERMINATED BY '\r\n' IGNORE 1 LINES (duration, start_date, end_date,
    ↪  start_station_number, start_station, end_station_number,
    ↪ end_station, bike_number, member_type);"
done;
```

3.1.4 Read Historical Data

This document describes how TiDB reads data from the history versions, how TiDB manages the data versions, as well as an example to show how to use the feature.

3.1.4.1 Feature description

TiDB implements a feature to read history data using the standard SQL interface directly without special clients or drivers. By using this feature:

- Even when data is updated or removed, its history versions can be read using the SQL interface.
- Even if the table structure changes after the data is updated, TiDB can use the old structure to read the history data.

3.1.4.2 How TiDB reads data from history versions

The `tidb_snapshot` system variable is introduced to support reading history data. About the `tidb_snapshot` variable:

- The variable is valid in the `Session` scope.
- Its value can be modified using the `Set` statement.
- The data type for the variable is text.
- The variable accepts TSO (Timestamp Oracle) and datetime. TSO is a globally unique time service, which is obtained from PD. The acceptable datetime format is "2016-10-08 16:45:26.999". Generally, the datetime can be set using second precision, for example "2016-10-08 16:45:26".

- When the variable is set, TiDB creates a Snapshot using its value as the timestamp, just for the data structure and there is no any overhead. After that, all the `Select` operations will read data from this Snapshot.

> **Note:**
>
> Because the timestamp in TiDB transactions is allocated by Placement Driver (PD), the version of the stored data is also marked based on the timestamp allocated by PD. When a Snapshot is created, the version number is based on the value of the `tidb_snapshot` variable. If there is a large difference between the local time of the TiDB server and the PD server, use the time of the PD server.

After reading data from history versions, you can read data from the latest version by ending the current Session or using the `Set` statement to set the value of the `tidb_snapshot` variable to "" (empty string).

3.1.4.3 How TiDB manages the data versions

TiDB implements Multi-Version Concurrency Control (MVCC) to manage data versions. The history versions of data are kept because each update/removal creates a new version of the data object instead of updating/removing the data object in-place. But not all the versions are kept. If the versions are older than a specific time, they will be removed completely to reduce the storage occupancy and the performance overhead caused by too many history versions.

In TiDB, Garbage Collection (GC) runs periodically to remove the obsolete data versions. For GC details, see TiDB Garbage Collection (GC)

Pay special attention to the following two variables:

- `tikv_gc_life_time`: It is used to configure the retention time of the history version. You can modify it manually.
- `tikv_gc_safe_point`: It records the current `safePoint`. You can safely create the snapshot to read the history data using the timestamp that is later than `safePoint`. `safePoint` automatically updates every time GC runs.

3.1.4.4 Example

1. At the initial stage, create a table and insert several rows of data:

```
create table t (c int);
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
insert into t values (1), (2), (3);
```

```
Query OK, 3 rows affected (0.00 sec)
```

2. View the data in the table:

```
select * from t;
```

```
+------+
| c    |
+------+
|    1 |
|    2 |
|    3 |
+------+
3 rows in set (0.00 sec)
```

3. View the timestamp of the table:

```
select now();
```

```
+---------------------+
| now()               |
+---------------------+
| 2016-10-08 16:45:26 |
+---------------------+
1 row in set (0.00 sec)
```

4. Update the data in one row:

```
update t set c=22 where c=2;
```

```
Query OK, 1 row affected (0.00 sec)
```

5. Make sure the data is updated:

```
select * from t;
```

```
+------+
| c    |
+------+
|    1 |
```

```
|   22 |
|    3 |
+------+
3 rows in set (0.00 sec)
```

6. Set the `tidb_snapshot` variable whose scope is Session. The variable is set so that the latest version before the value can be read.

> **Note:**
> In this example, the value is set to be the time before the update operation.

```
set @@tidb_snapshot="2016-10-08 16:45:26";
```

```
Query OK, 0 rows affected (0.00 sec)
```

> **Note:**
> You should use `@@` instead of `@` before `tidb_snapshot` because `@@` is used to denote the system variable while `@` is used to denote the user variable.

Result: The read from the following statement is the data before the update operation, which is the history data.

```
select * from t;
```

```
+------+
| c    |
+------+
|    1 |
|    2 |
|    3 |
+------+
3 rows in set (0.00 sec)
```

7. Set the `tidb_snapshot` variable to be "" (empty string) and you can read the data from the latest version:

```
set @@tidb_snapshot="";
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
select * from t;
```

```
+------+
| c    |
+------+
|    1 |
|   22 |
|    3 |
+------+
3 rows in set (0.00 sec)
```

> **Note:**
> You should use `@@` instead of `@` before `tidb_snapshot` because `@@` is used to denote the system variable while `@` is used to denote the user variable.

3.1.5 TiDB Binlog Tutorial

This tutorial starts with a simple TiDB Binlog deployment with a single node of each component (Placement Driver, TiKV Server, TiDB Server, Pump, and Drainer), set up to push data into a MariaDB Server instance.

This tutorial is targeted toward users who have some familiarity with the TiDB Architecture, who may have already set up a TiDB cluster (not mandatory), and who wants to gain hands-on experience with TiDB Binlog. This tutorial is a good way to "kick the tires" of TiDB Binlog and to familiarize yourself with the concepts of its architecture.

> **Warning:**
> The instructions to deploy TiDB in this tutorial should **not** be used to deploy TiDB in a production or development setting.

This tutorial assumes you're using a modern Linux distribution on x86-64. A minimal CentOS 7 installation running in VMware is used in this tutorial for the examples. It's recommended that you start from a clean install, so that you aren't impacted by quirks of your existing environment. If you don't want to use local virtualization, you can easily start a CentOS 7 VM using your cloud service.

3.1.5.1 TiDB Binlog Overview

TiDB Binlog is a solution to collect binary log data from TiDB and provide real-time data backup and replication. It pushes incremental data updates from a TiDB Server cluster into downstream platforms.

You can use TiDB Binlog for incremental backups, to replicate data from one TiDB cluster to another, or to send TiDB updates through Kafka to a downstream platform of your choice.

TiDB Binlog is particularly useful when you migrate data from MySQL or MariaDB to TiDB, in which case you may use the TiDB DM (Data Migration) platform to get data from a MySQL/MariaDB cluster into TiDB, and then use TiDB Binlog to keep a separate, downstream MySQL/MariaDB instance/cluster in sync with your TiDB cluster. TiDB Binlog enables application traffic to TiDB to be pushed to a downstream MySQL or MariaDB instance/cluster, which reduces the risk of a migration to TiDB because you can easily revert the application to MySQL or MariaDB without downtime or data loss.

See TiDB Binlog Cluster User Guide for more information.

3.1.5.2 Architecture

TiDB Binlog comprises two components: the **Pump** and the **Drainer**. Several Pump nodes make up a pump cluster. Each Pump node connects to TiDB Server instances and receives updates made to each of the TiDB Server instances in a cluster. A Drainer connects to the Pump cluster and transforms the received updates into the correct format for a particular downstream destination, for example, Kafka, another TiDB Cluster or a MySQL/MariaDB server.

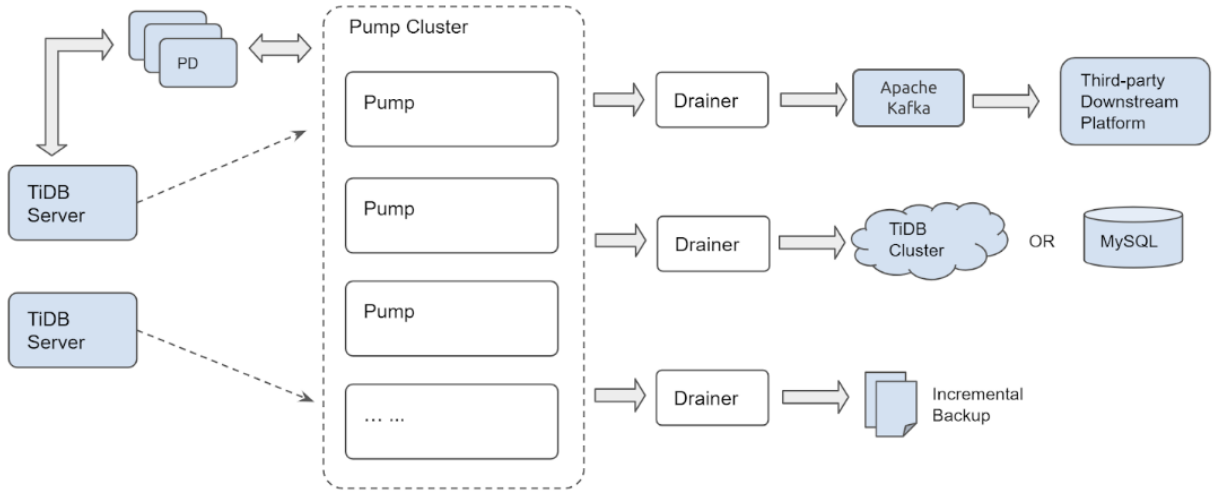

Figure 24: TiDB-Binlog architecture

The clustered architecture of Pump ensures that updates won't be lost as new TiDB

Server instances join or leave the TiDB Cluster or Pump nodes join or leave the Pump cluster.

3.1.5.3 Installation

We're using MariaDB Server in this case instead of MySQL Server because RHEL/CentOS 7 includes MariaDB Server in their default package repositories. We'll need the client as well as the server for later use. Let's install them now:

```
sudo yum install -y mariadb-server
```

Even if you've already started a TiDB cluster, it will be easier to follow along with this tutorial where we will set up a new, simple cluster. We will install from a tarball, using a simplified form of the Local Deployment guide. You may also wish to refer to Testing Deployment from Binary Tarball for best practices of establishing a real testing deployment, but that goes beyond the scope of this tutorial.

```
curl -L https://download.pingcap.org/tidb-v3.0-linux-amd64.tar.gz | tar xzf
    ↪ - &&
cd tidb-v3.0-linux-amd64/
```

Expected output:

```
[kolbe@localhost ~]$ curl -LO https://download.pingcap.org/tidb-v3.0-linux-
    ↪ amd64.tar.gz | tar xzf -
  % Total    % Received % Xferd Average Speed Time  Time    Time Current
                                Dload Upload  Total  Spent   Left Speed
100  279M 100  279M   0     0  4983k     0  0:00:57 0:00:57 --:--:-- 3638k
[kolbe@localhost ~]$ cd tidb-v3.0-linux-amd64/
[kolbe@localhost tidb-v3.0-linux-amd64]$
```

3.1.5.4 Configuration

Now we'll start a simple TiDB cluster, with a single instance for each of `pd-server`, `tikv-server`, and `tidb-server`.

Populate the config files using:

```
printf > pd.toml %s\\n 'log-file="pd.log"' 'data-dir="pd.data"' &&
printf > tikv.toml %s\\n 'log-file="tikv.log"' '[storage]' 'data-dir="tikv.
    ↪ data"' '[pd]' 'endpoints=["127.0.0.1:2379"]' '[rocksdb]' max-open-
    ↪ files=1024 '[raftdb]' max-open-files=1024 &&
printf > pump.toml %s\\n 'log-file="pump.log"' 'data-dir="pump.data"' 'addr
    ↪ ="127.0.0.1:8250"' 'advertise-addr="127.0.0.1:8250"' 'pd-urls="http
    ↪ ://127.0.0.1:2379"' &&
printf > tidb.toml %s\\n 'store="tikv"' 'path="127.0.0.1:2379"' '[log.file
    ↪ ]' 'filename="tidb.log"' '[binlog]' 'enable=true' &&
```

```
printf > drainer.toml %s\\n 'log-file="drainer.log"' '[syncer]' 'db-type="
    ↪ mysql"' '[syncer.to]' 'host="127.0.0.1"' 'user="root"' 'password=""'
    ↪ 'port=3306'
```

Use the following commands to see the configuration details:

```
for f in *.toml; do echo "$f:"; cat "$f"; echo; done
```

Expected output:

```
drainer.toml:
log-file="drainer.log"
[syncer]
db-type="mysql"
[syncer.to]
host="127.0.0.1"
user="root"
password=""
port=3306

pd.toml:
log-file="pd.log"
data-dir="pd.data"

pump.toml:
log-file="pump.log"
data-dir="pump.data"
addr="127.0.0.1:8250"
advertise-addr="127.0.0.1:8250"
pd-urls="http://127.0.0.1:2379"

tidb.toml:
store="tikv"
path="127.0.0.1:2379"
[log.file]
filename="tidb.log"
[binlog]
enable=true

tikv.toml:
log-file="tikv.log"
[storage]
data-dir="tikv.data"
[pd]
endpoints=["127.0.0.1:2379"]
[rocksdb]
```

```
max-open-files=1024
[raftdb]
max-open-files=1024
```

3.1.5.5 Bootstrapping

Now we can start each component. This is best done in a specific order - firstly the Placement Driver (PD), then TiKV Server, then Pump (because TiDB must connect to the Pump service to send the binary log), and finally the TiDB Server.

Start all the services using:

```
./bin/pd-server --config=pd.toml &>pd.out &
./bin/tikv-server --config=tikv.toml &>tikv.out &
./bin/pump --config=pump.toml &>pump.out &
sleep 3
./bin/tidb-server --config=tidb.toml &>tidb.out &
```

Expected output:

```
[kolbe@localhost tidb-v3.0-linux-amd64]$ ./bin/pd-server --config=pd.toml &>
   ↪ pd.out &
[1] 20935
[kolbe@localhost tidb-v3.0-linux-amd64]$ ./bin/tikv-server --config=tikv.
   ↪ toml &>tikv.out &
[2] 20944
[kolbe@localhost tidb-v3.0-linux-amd64]$ ./bin/pump --config=pump.toml &>
   ↪ pump.out &
[3] 21050
[kolbe@localhost tidb-v3.0-linux-amd64]$ sleep 3
[kolbe@localhost tidb-v3.0-linux-amd64]$ ./bin/tidb-server --config=tidb.
   ↪ toml &>tidb.out &
[4] 21058
```

If you execute `jobs`, you should see a list of running daemons:

```
jobs
```

```
[1]   Running                 ./bin/pd-server --config=pd.toml &>pd.out &
[2]   Running                 ./bin/tikv-server --config=tikv.toml &>tikv.out &
[3]-  Running                 ./bin/pump --config=pump.toml &>pump.out &
[4]+  Running                 ./bin/tidb-server --config=tidb.toml &>tidb.out &
```

If one of the services has failed to start (if you see "`Exit 1`" instead of "`Running`", for example), try to restart that individual service.

3.1.5.6 Connecting

You should have all 4 components of our TiDB Cluster running now, and you can now connect to the TiDB Server on port 4000 using the MariaDB/MySQL command-line client:

```
mysql -h 127.0.0.1 -P 4000 -u root -e 'select tidb_version();'
```

Expected output:

```
*************************** 1. row ***************************
tidb_version(): Release Version: v3.0.0
Git Commit Hash: 60965b006877ca7234adaced7890d7b029ed1306
Git Branch: HEAD
UTC Build Time: 2019-06-28 12:14:07
GoVersion: go version go1.12 linux/amd64
Race Enabled: false
TiKV Min Version: 2.1.0-alpha.1-ff3dd160846b7d1aed9079c389fc188f7f5ea13e
Check Table Before Drop: false
```

At this point we have a TiDB Cluster running, and we have `pump` reading binary logs from the cluster and storing them as relay logs in its data directory. The next step is to start a MariaDB server that `drainer` can write to.

Start `drainer` using:

```
sudo systemctl start mariadb &&
./bin/drainer --config=drainer.toml &>drainer.out &
```

If you are using an operating system that makes it easier to install MySQL server, that's also OK. Just make sure it's listening on port 3306 and that you can either connect to it as user "root" with an empty password, or adjust drainer.toml as necessary.

```
mysql -h 127.0.0.1 -P 3306 -u root
```

Expected output:

```
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 20
Server version: 5.5.60-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
   ↪ statement.

MariaDB [(none)]>
```

```
show databases;
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
| tidb_binlog        |
+--------------------+
5 rows in set (0.01 sec)
```

Here we can already see the `tidb_binlog` database, which contains the `checkpoint` table used by `drainer` to record up to what point binary logs from the TiDB cluster have been applied.

```
use tidb_binlog;
```

```
Database changed
```

```
select * from checkpoint;
```

```
+--------------------+-------------------------------------------------+
| clusterID          | checkPoint                                      |
+--------------------+-------------------------------------------------+
| 6678715361817107733 | {"commitTS":407637466476445697,"ts-map":{}} |
+--------------------+-------------------------------------------------+
1 row in set (0.00 sec)
```

Now, let's open another client connection to the TiDB server, so that we can create a table and insert some rows into it. (It's recommended that you do this under a GNU screen so you can keep multiple clients open at the same time.)

```
mysql -h 127.0.0.1 -P 4000 --prompt='TiDB [\d]> ' -u root
```

```
create database tidbtest;
use tidbtest;
create table t1 (id int unsigned not null AUTO_INCREMENT primary key);
insert into t1 () values (),(),(),(),();
select * from t1;
```

Expected output:

```
TiDB [(none)]> create database tidbtest;
Query OK, 0 rows affected (0.12 sec)
```

```
TiDB [(none)]> use tidbtest;
Database changed
TiDB [tidbtest]> create table t1 (id int unsigned not null AUTO_INCREMENT
    ↪ primary key);
Query OK, 0 rows affected (0.11 sec)

TiDB [tidbtest]> insert into t1 () values (),(),(),(),();
Query OK, 5 rows affected (0.01 sec)
Records: 5 Duplicates: 0 Warnings: 0

TiDB [tidbtest]> select * from t1;
+----+
| id |
+----+
|  1 |
|  2 |
|  3 |
|  4 |
|  5 |
+----+
5 rows in set (0.00 sec)
```

Switching back to the MariaDB client, we should find the new database, new table, and the newly inserted rows:

```
use tidbtest;
show tables;
select * from t1;
```

Expected output:

```
MariaDB [(none)]> use tidbtest;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [tidbtest]> show tables;
+--------------------+
| Tables_in_tidbtest |
+--------------------+
| t1                 |
+--------------------+
1 row in set (0.00 sec)

MariaDB [tidbtest]> select * from t1;
+----+
```

```
| id |
+----+
|  1 |
|  2 |
|  3 |
|  4 |
|  5 |
+----+
5 rows in set (0.00 sec)
```

You should see the same rows that you inserted into TiDB when querying the MariaDB server. Congratulations! You've just set up TiDB Binlog!

3.1.5.7 binlogctl

Information about Pumps and Drainers that have joined the cluster is stored in PD. You can use the binlogctl tool query and manipulate information about their states. See binlogctl guide for more information.

Use `binlogctl` to get a view of the current status of Pumps and Drainers in the cluster:

```
./bin/binlogctl -cmd drainers &&
./bin/binlogctl -cmd pumps
```

Expected output:

```
[kolbe@localhost tidb-v3.0-linux-amd64]$ ./bin/binlogctl -cmd drainers
[2019/04/11 17:44:10.861 -04:00] [INFO] [nodes.go:47] ["query node"] [type=
    ↪ drainer] [node="{NodeID: localhost.localdomain:8249, Addr:
    ↪ 192.168.236.128:8249, State: online, MaxCommitTS: 407638907719778305,
    ↪  UpdateTime: 2019-04-11 17:44:10 -0400 EDT}"]

[kolbe@localhost tidb-v3.0-linux-amd64]$ ./bin/binlogctl -cmd pumps
[2019/04/11 17:44:13.904 -04:00] [INFO] [nodes.go:47] ["query node"] [type=
    ↪ pump] [node="{NodeID: localhost.localdomain:8250, Addr:
    ↪ 192.168.236.128:8250, State: online, MaxCommitTS: 407638914024079361,
    ↪  UpdateTime: 2019-04-11 17:44:13 -0400 EDT}"]
```

If you kill a Drainer, the cluster puts it in the "paused" state, which means that the cluster expects it to rejoin:

```
pkill drainer &&
./bin/binlogctl -cmd drainers
```

Expected output:

```
[kolbe@localhost tidb-v3.0-linux-amd64]$ pkill drainer
[kolbe@localhost tidb-v3.0-linux-amd64]$ ./bin/binlogctl -cmd drainers
```

```
[2019/04/11 17:44:22.640 -04:00] [INFO] [nodes.go:47] ["query node"] [type=
    ↪ drainer] [node="{NodeID: localhost.localdomain:8249, Addr:
    ↪ 192.168.236.128:8249, State: paused, MaxCommitTS: 407638915597467649,
    ↪  UpdateTime: 2019-04-11 17:44:18 -0400 EDT}"]
```

You can use "NodeIDs" with `binlogctl` to control individual nodes. In this case, the NodeID of the drainer is "localhost.localdomain:8249" and the NodeID of the Pump is "localhost.localdomain:8250".

The main use of `binlogctl` in this tutorial is likely to be in the event of a cluster restart. If you end all processes in the TiDB cluster and try to restart them (not including the downstream MySQL/MariaDB server or Drainer), Pump will refuse to start because it cannot contact Drainer and believe that Drainer is still "online".

There are 3 solutions to this issue:

- Stop Drainer using `binlogctl` instead of killing the process:

```
./bin/binlogctl --pd-urls=http://127.0.0.1:2379 --cmd=drainers &&
./bin/binlogctl --pd-urls=http://127.0.0.1:2379 --cmd=offline-drainer
    ↪ --node-id=localhost.localdomain:8249
```

- Start Drainer *before* starting Pump.
- Use `binlogctl` after starting PD (but before starting Drainer and Pump) to update the state of the paused Drainer:

```
./bin/binlogctl --pd-urls=http://127.0.0.1:2379 --cmd=update-drainer --
    ↪ node-id=localhost.localdomain:8249 --state=offline
```

3.1.5.8 Cleanup

To stop the TiDB cluster and TiDB Binlog processes, you can execute `pkill -P $$` in the shell where you started all the processes that form the cluster (pd-server, tikv-server, pump, tidb-server, drainer). To give each component enough time to shut down cleanly, it's helpful to stop them in a particular order:

```
for p in tidb-server drainer pump tikv-server pd-server; do pkill "$p";
    ↪ sleep 1; done
```

Expected output:

```
kolbe@localhost tidb-v3.0-linux-amd64]$ for p in tidb-server drainer pump
    ↪ tikv-server pd-server; do pkill "$p"; sleep 1; done
[4]- Done                    ./bin/tidb-server --config=tidb.toml &>tidb.out
[5]+ Done                    ./bin/drainer --config=drainer.toml &>drainer.out
[3]+ Done                    ./bin/pump --config=pump.toml &>pump.out
[2]+ Done                    ./bin/tikv-server --config=tikv.toml &>tikv.out
[1]+ Done                    ./bin/pd-server --config=pd.toml &>pd.out
```

If you wish to restart the cluster after all services exit, use the same commands you ran originally to start the services. As discussed in the binlogctl section above, you'll need to start `drainer` before `pump`, and `pump` before `tidb-server`.

```
./bin/pd-server --config=pd.toml &>pd.out &
./bin/tikv-server --config=tikv.toml &>tikv.out &
./bin/drainer --config=drainer.toml &>drainer.out &
sleep 3
./bin/pump --config=pump.toml &>pump.out &
sleep 3
./bin/tidb-server --config=tidb.toml &>tidb.out &
```

If any of the components fail to start, try to restart the failed individual component(s).

3.1.5.9 Conclusion

In this tutorial, we've set up TiDB Binlog to replicate from a TiDB cluster to a downstream MariaDB server, using a cluster with a single Pump and a single Drainer. As we've seen, TiDB Binlog is a comprehensive platform for capturing and processing changes to a TiDB cluster.

In a more robust development, testing, or production deployment, you'd have multiple TiDB servers for high availability and scaling purposes, and you'd use multiple Pump instances to ensure that application traffic to TiDB server instances is unaffected by problems in the Pump cluster. You may also use additional Drainer instances to push updates to different downstream platforms or to implement incremental backups.

3.1.6 TiDB Data Migration Tutorial

3.1.7 TiDB Lightning Tutorial

TiDB Lightning is a tool used for fast full import of large amounts of data into a TiDB cluster. Currently, TiDB Lightning supports reading SQL dump exported via Mydumper or CSV data source. You can use it in the following two scenarios:

- Import **large amounts** of **new** data **quickly**
- Back up and restore all the data

The TiDB Lightning tool set consists of two components:

- **`tidb-lightning`** (the "front end") reads the data source and imports the database structure into the TiDB cluster, and also transforms the data into Key-Value (KV) pairs and sends them to `tikv-importer`.
- **`tikv-importer`** (the "back end") combines and sorts the KV pairs and then imports these sorted pairs as a whole into the TiKV cluster.

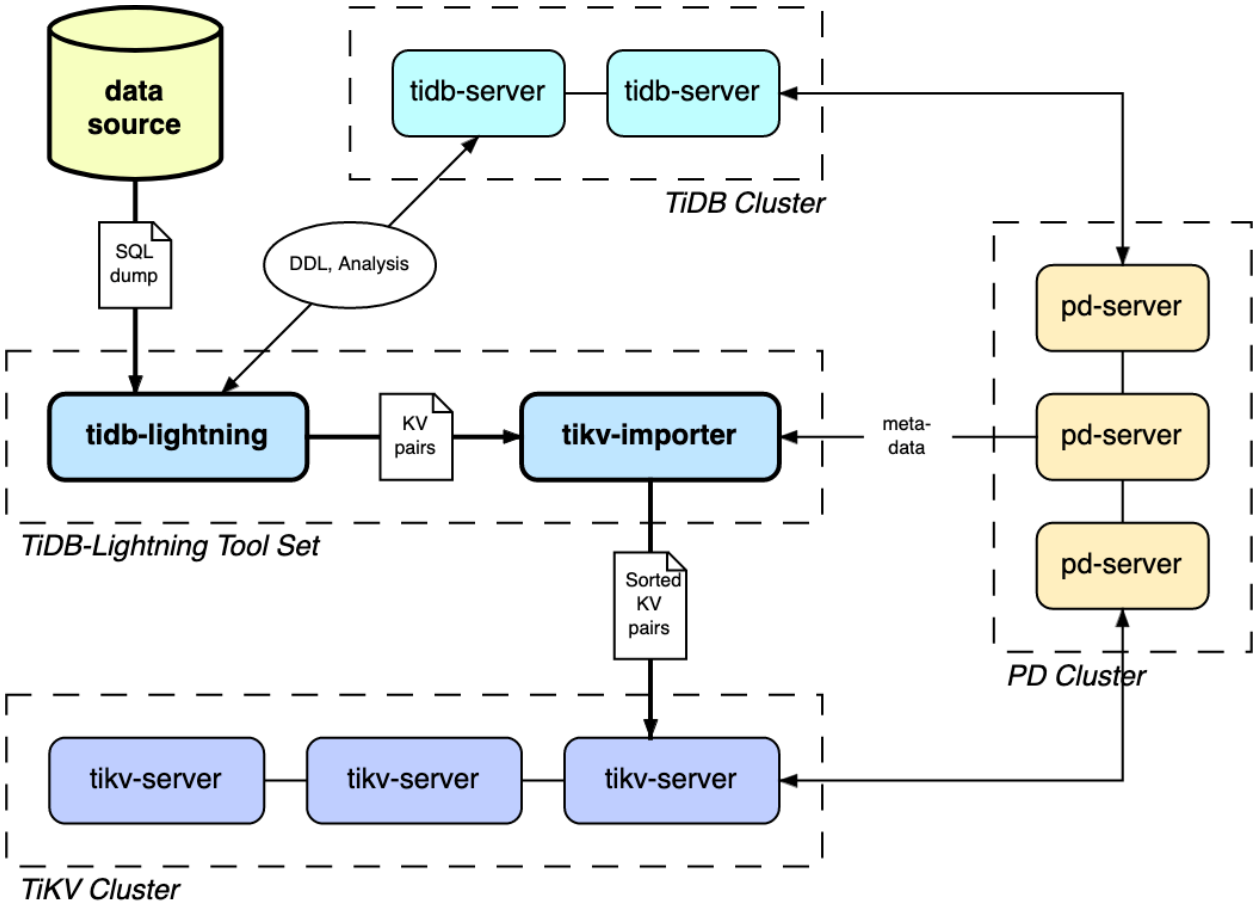

Figure 25: Architecture of TiDB Lightning tool set

3.1.7.1 Prerequisites

This tutorial assumes you use several new and clean CentOS 7 instances. You can use VMware, VirtualBox or other tools to deploy a virtual machine locally or a small cloud virtual machine on a vendor-supplied platform. Because TiDB Lightning consumes a large amount of computer resources, it is recommended that you allocate at least 4 GB memory for running it.

> **Warning:**
>
> The deployment method in this tutorial is only recommended for test and trial. **Do not apply it in the production or development environment.**

3.1.7.2 Prepare full backup data

First, use mydumper to export data from MySQL:

```
./bin/mydumper -h 127.0.0.1 -P 3306 -u root -t 16 -F 256 -B test -T t1,t2
    ↪ --skip-tz-utc -o /data/my_database/
```

In the above command:

- `-B test`: means the data is exported from the `test` database.
- `-T t1,t2`: means only the `t1` and `t2` tables are exported.
- `-t 16`: means 16 threads are used to export the data.
- `-F 256`: means a table is partitioned into chunks and one chunk is 256 MB.
- `--skip-tz-utc`: the purpose of adding this parameter is to ignore the inconsistency of time zone setting between MySQL and the data exporting machine and to disable automatic conversion.

After executing this command, the full backup data is exported to the `/data/my_database` directory.

3.1.7.3 Deploy TiDB Lightning

3.1.7.3.1 Step 1: Deploy TiDB cluster

Before the data import, you need to deploy a TiDB cluster (later than v2.0.9). In this tutorial, TiDB v3.0.4 is used. For the deployment method, refer to TiDB Introduction.

3.1.7.3.2 Step 2: Download TiDB Lightning installation package

Download the TiDB Lightning installation package from the following link:

- **v3.0.4**: tidb-toolkit-v3.0.4-linux-amd64.tar.gz

> **Note:**
>
> Choose the same version of TiDB Lightning as that of the TiDB cluster.

3.1.7.3.3 Step 3: Start `tikv-importer`

1. Upload `bin/tikv-importer` in the package to the server where TiDB Lightning is deployed.
2. Configure `tikv-importer.toml`.

```
# The template of the tikv-importer configuration file

# Log file
log-file = "tikv-importer.log"
# Log level: "trace", "debug", "info", "warn", "error" or "off"
log-level = "info"

[server]
# The listening address of tikv-importer. tidb-lightning connects to
   ↪ this address for data write.
addr = "192.168.20.10:8287"

[import]
# The directory of the engine file.
import-dir = "/mnt/ssd/data.import/"
```

3. Run `tikv-importer`:

```
nohup ./tikv-importer -C tikv-importer.toml > nohup.out &
```

3.1.7.3.4 Step 4: Start `tidb-lightning`

1. Upload `bin/tidb-lightning` and `bin/tidb-lightning-ctl` in the installation package to the server where TiDB Lightning is deployed.

2. Upload the prepared data source to the server.

3. After configuring the parameters properly, use a `nohup` command to start the `tidb ↪ -lightning` process. If you directly run the command in the command-line, the process might exit because of the SIGHUP signal received. Instead, it's preferable to run a bash script that contains the `nohup` command:

```
#!/bin/bash
nohup ./tidb-lightning \
          --importer 172.16.31.10:8287 \
          -d /data/my_database/ \
          --tidb-host 172.16.31.2 \
          --tidb-user root \
          --log-file tidb-lightning.log \
      > nohup.out &
```

3.1.7.3.5 Step 5: Check data integrity

After the import is completed, TiDB Lightning exits automatically. If the import is successful, you can find `tidb lightning exit` in the last line of the log file.

If any error occurs, refer to TiDB Lightning FAQs.

3.1.7.4 Summary

This tutorial briefly introduces what TiDB Lightning is and how to quickly deploy a TiDB Lightning cluster to import full backup data to the TiDB cluster.

For detailed features and usage about TiDB Lightning, refer to TiDB Lightning Overview.

3.1.8 TiSpark Quick Start Guide

To make it easy to try TiSpark, the TiDB cluster installed using TiDB Ansible integrates Spark, TiSpark jar package and TiSpark sample data by default.

3.1.8.1 Deployment information

- Spark is deployed by default in the `spark` folder in the TiDB instance deployment directory.
- The TiSpark jar package is deployed by default in the `jars` folder in the Spark deployment directory.

```
spark/jars/tispark-${name_with_version}.jar
```

- TiSpark sample data and import scripts can be downloaded from TiSpark sample data.

```
tispark-sample-data/
```

3.1.8.2 Prepare the environment

3.1.8.2.1 Install JDK on the TiDB instance

Download the latest version of JDK 1.8 from Oracle JDK official download page. The version used in the following example is `jdk-8u141-linux-x64.tar.gz`.

Extract the package and set the environment variables based on your JDK deployment directory.

Edit the `~/.bashrc` file. For example:

```
export JAVA_HOME=/home/pingcap/jdk1.8.0_144 &&
export PATH=$JAVA_HOME/bin:$PATH
```

Verify the validity of JDK:

```
java -version
```

```
java version "1.8.0_144"
Java(TM) SE Runtime Environment (build 1.8.0_144-b01)
Java HotSpot(TM) 64-Bit Server VM (build 25.144-b01, mixed mode)
```

3.1.8.2.2 Import the sample data

Assume that the TiDB cluster is started. The service IP of one TiDB instance is `192.168.0.2`, the port is `4000`, the user name is `root`, and the password is null.

```
wget http://download.pingcap.org/tispark-sample-data.tar.gz
tar -zxvf tispark-sample-data.tar.gz
cd tispark-sample-data
```

Edit the TiDB login information in `sample_data.sh`. For example:

```
mysql --local-infile=1 -h 192.168.0.2 -P 4000 -u root < dss.ddl
```

Run the script:

```
./sample_data.sh
```

> **Note:**
>
> You need to install the MySQL client on the machine that runs the script. If you are a CentOS user, you can install it through the command `yum -y install mysql`.

Log into TiDB and verify that the `TPCH_001` database and the following tables are included.

```
mysql -uroot -P4000 -h192.168.0.2
```

```
show databases;
```

```
+--------------------+
| Database           |
+--------------------+
| INFORMATION_SCHEMA |
| PERFORMANCE_SCHEMA |
```

```
| TPCH_001           |
| mysql              |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

```
use TPCH_001;
```

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

```
show tables;
```

```
+--------------------+
| Tables_in_TPCH_001 |
+--------------------+
| CUSTOMER           |
| LINEITEM           |
| NATION             |
| ORDERS             |
| PART               |
| PARTSUPP           |
| REGION             |
| SUPPLIER           |
+--------------------+
8 rows in set (0.00 sec)
```

3.1.8.3 Use example

First start the spark-shell:

```
cd spark &&
bin/spark-shell
```

Then query the TiDB table as you are using the native Spark SQL:

```
scala> spark.sql("use TPCH_001")
scala> spark.sql("select count(*) from lineitem").show
```

The result is:

```
+--------+
|count(1)|
+--------+
```

```
|   60175|
+--------+
```

Now run a more complex Spark SQL:

```
scala> spark.sql(
     """select
       |   l_returnflag,
       |   l_linestatus,
       |   sum(l_quantity) as sum_qty,
       |   sum(l_extendedprice) as sum_base_price,
       |   sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
       |   sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as
         ↪ sum_charge,
       |   avg(l_quantity) as avg_qty,
       |   avg(l_extendedprice) as avg_price,
       |   avg(l_discount) as avg_disc,
       |   count(*) as count_order
       |from
       |   lineitem
       |where
       |   l_shipdate <= date '1998-12-01' - interval '90' day
       |group by
       |   l_returnflag,
       |   l_linestatus
       |order by
       |   l_returnflag,
       |   l_linestatus
     """.stripMargin).show
```

The result is:

```
+------------+------------+--------+--------------+--------------+
|l_returnflag|l_linestatus| sum_qty|sum_base_price|sum_disc_price|
+------------+------------+--------+--------------+--------------+
|           A|           F|380456.00| 532348211.65|505822441.4861|
|           N|           F| 8971.00|  12384801.37| 11798257.2080|
|           N|           O|742802.00| 1041502841.45|989737518.6346|
|           R|           F|381449.00| 534594445.35|507996454.4067|
+------------+------------+--------+--------------+--------------+
(Continued)
-----------------+---------+------------+--------+-----------+
      sum_charge| avg_qty| avg_price|avg_disc|count_order|
-----------------+---------+------------+--------+-----------+
 526165934.000839|25.575155|35785.709307|0.050081| 14876|
  12282485.056933|25.778736|35588.509684|0.047759| 348|
```

```
1029418531.523350|25.454988|35691.129209|0.049931| 29181|
 528524219.358903|25.597168|35874.006533|0.049828| 14902|
----------------+---------+------------+--------+-----------+
```

See more examples.

3.2 Deploy

3.2.1 Software and Hardware Recommendations

3.2.1.1 About

As an open source distributed NewSQL database with high performance, TiDB can be deployed in the Intel architecture server and major virtualization environments and runs well. TiDB supports most of the major hardware networks and Linux operating systems.

3.2.1.2 Linux OS version requirements

Linux OS Platform	Version
Red Hat Enterprise Linux	7.3 or later
CentOS	7.3 or later
Oracle Enterprise Linux	7.3 or later
Ubuntu LTS	16.04 or later

Note:

- For Oracle Enterprise Linux, TiDB supports the Red Hat Compatible Kernel (RHCK) and does not support the Unbreakable Enterprise Kernel provided by Oracle Enterprise Linux.
- A large number of TiDB tests have been run on the CentOS 7.3 system, and in our community there are a lot of best practices in which TiDB is deployed on the Linux operating system. Therefore, it is recommended to deploy TiDB on CentOS 7.3 or later.
- The support for the Linux operating systems above includes the deployment and operation in physical servers as well as in major virtualized environments like VMware, KVM and XEN.

3.2.1.3 Server recommendations

You can deploy and run TiDB on the 64-bit generic hardware server platform in the Intel

x86-64 architecture. The requirements and recommendations about server hardware configuration (ignoring the resources occupied by the operating system itself) for development, test, and production environments are as follows:

3.2.1.3.1 Development and test environments

Component	CPU	Memory	Local Storage	Network	Instance Number (Minimum Requirement)
TiDB	8 core+	16 GB+	No special requirements	Gigabit network card	1 (can be deployed on the same machine with PD)
PD	4 core+	8 GB+	SAS, 200 GB+	Gigabit network card	1 (can be deployed on the same machine with TiDB)
TiKV	8 core+	32 GB+	SAS, 200 GB+	Gigabit network card	3
				Total Server Number	4

Note:

- In the test environment, the TiDB and PD instances can be deployed on the same server.
- For performance-related test, do not use low-performance storage and network hardware configuration, in order to guarantee the correctness of the test result.
- For the TiKV server, it is recommended to use NVMe SSDs to ensure faster reads and writes.
- The TiDB server uses the disk to store server logs, so there are no special requirements for the disk type and capacity in the test environment.

3.2.1.3.2 Production environment

Component	CPU	Memory	Hard Disk Type	Network	Instance Nu
TiDB	16 core+	32 GB+	SAS	10 Gigabit network card (2 preferred)	

Component	CPU	Memory	Hard Disk Type	Network	Instance Nu
PD	4 core+	8 GB+	SSD	10 Gigabit network card (2 preferred)	
TiKV	16 core+	32 GB+	SSD	10 Gigabit network card (2 preferred)	
Monitor	8 core+	16 GB+	SAS	Gigabit network card	
				Total Server Number	

Note:

- In the production environment, the TiDB and PD instances can be deployed on the same server. If you have a higher requirement for performance and reliability, try to deploy them separately.
- It is strongly recommended to use higher configuration in the production environment.
- It is recommended to keep the size of TiKV hard disk within 2 TB if you are using PCIe SSDs or within 1.5 TB if you are using regular SSDs.

3.2.1.4 Network requirements

As an open source distributed NewSQL database, TiDB requires the following network port configuration to run. Based on the TiDB deployment in actual environments, the administrator can open relevant ports in the network side and host side.

Component	Default Port	Description
TiDB	4000	the communication port for the application and DBA tools

Component	Default Port	Description
TiDB	10080	the communication port to report TiDB status
TiKV	20160	the TiKV communication port
TiKV	20180	the communication port to report TiKV status
PD	2379	the communication port between TiDB and PD

Component	Default Port	Description
PD	2380	the inter-node com-mu-nica-tion port within the PD clus-ter
Pump	8250	the Pump com-mu-nica-tion port
Drainer	8249	the Drainer com-mu-nica-tion port
Prometheus	9090	the com-mu-nica-tion port for the Prometheus ser-vice

Component	Default Port	Description
Pushgateway	9091	the aggregation and report port for tikv-importer
Node_exporter	9100	the communication port to report the system information of every TiDB cluster node

Component	Default Port	Description
Blackbox_exporter	9115	the Blackbox_exporter communication port, used to monitor the ports in the TiDB cluster
Grafana	3000	the port for the external Web monitoring service and client (Browser) access

Component	Default Port	Description
Grafana	8686	the grafana_collector communication port, used to export the Dashboard as the PDF format
Kafka_exporter	9308	the Kafka_exporter communication port, used to monitor the binlog Kafka cluster

3.2.1.5 Web browser requirements

TiDB relies on Grafana to provide visualization of database metrics. A recent version of Internet Explorer, Chrome or Firefox with Javascript enabled is sufficient.

3.2.2 From Binary Tarball

3.2.2.1 Testing Deployment from Binary Tarball

This guide provides installation instructions for all TiDB components across multiple nodes for testing purposes. It does not match the recommended usage for production systems.

See also local deployment and production environment deployment.

3.2.2.1.1 Prepare

Before you start, see TiDB architecture and Software and Hardware Recommendations. Make sure the following requirements are satisfied:

Operating system

For the operating system, it is recommended to use RHEL/CentOS 7.3 or higher. The following additional requirements are recommended:

Configuration	Description
Supported Platform	RHEL/CentOS 7.3+ (more details)
File System	ext4 is recommended
Swap Space	Should be disabled
Disk Block Size	Set the system disk `Block` size to `4096`

Network and firewall

Configuration	Description
Firewall/Port	Check whether the ports required by TiDB are accessible between the nodes

Operating system parameters

Configuration	Description
Nice Limits	For system users, set the default value of `nice` in TiDB to `0`
min_free_kbytes	The setting for `vm.min_free_kbytes` in `sysctl.conf` needs to be high enough

Configuration	Description
User Open Files Limit	For database administrators, set the number of TiDB open files to `1000000`
System Open File Limits	Set the number of system open files to `1000000`
User Process Limits	For TiDB users, set the `nproc` value to `4096` in `limits.conf`
Address Space Limits	For TiDB users, set the space to `unlimited` in `limits.conf`
File Size Limits	For TiDB users, set the `fsize` value to `unlimited` in `limits.conf`
Disk Readahead	Set the value of the `readahead` data disk to `4096` at a minimum
NTP service	Configure the NTP time synchronization service for each node
SELinux	Turn off the SELinux service for each node
CPU Frequency Scaling	It is recommended to turn on CPU overclocking
Transparent Hugepages	For Red Hat 7+ and CentOS 7+ systems, it is required to set the Transparent Hugepages to `always`
I/O Scheduler	Set the I/O Scheduler of data disks to the `deadline` mode

Configuration	Description
vm.swappiness	Set `vm.swappiness = 0` in `sysctl.conf`
net.core.somaxconn	Set `net.core.somaxconn = 32768` in `sysctl.conf`
net.ipv4.tcp_syncookies	Set `net.ipv4.tcp_syncookies = 0` in `sysctl.conf`

Database running user settings

Configuration	Description
LANG environment	Set `LANG = en_US.UTF8`
TZ time zone	Set the TZ time zone of all nodes to the same value

3.2.2.1.2 TiDB components and default ports

Before you deploy a TiDB cluster, see the required components and optional components.

TiDB database components (required)

See the following table for the default ports for the TiDB components:

Component	Default Port	Protocol	Description
ssh	22	TCP	the sshd service
TiDB	4000	TCP	the communication port for the application and DBA tools
TiDB	10080	TCP	the communication port to report TiDB status
TiKV	20160	TCP	the TiKV communication port
PD	2379	TCP	the communication port between TiDB and PD
PD	2380	TCP	the inter-node communication port within the PD cluster

TiDB database components (optional)

See the following table for the default ports for the optional TiDB components:

Component	Default Port	Protocol	Description
Prometheus	9090	TCP	the communication port for the Prometheus service
Pushgateway	9091	TCP	the aggregation and report port for TiDB, TiKV, and PD monitor

Component	Default Port	Protocol	Description
Node_exporter	9100	TCP	the communication port to report the system information of every TiDB cluster node
Grafana	3000	TCP	the port for the external Web monitoring service and client (Browser) access
alertmanager	9093	TCP	the port for the alert service

3.2.2.1.3 Create a database running user account

1. Log in to the machine using the `root` user account and create a database running user account (`tidb`) using the following command:

```
useradd tidb -m
```

2. Switch the user from `root` to `tidb` by using the following command. You can use this `tidb` user account to deploy your TiDB cluster.

```
su - tidb
```

3.2.2.1.4 Download the official binary package

1. Download the package.

```
wget https://download.pingcap.org/tidb-v3.0-linux-amd64.tar.gz https://
    ↪ download.pingcap.org/tidb-v3.0-linux-amd64.sha256
```

2. Check the file integrity. If the result is OK, the file is correct.

```
sha256sum -c tidb-v3.0-linux-amd64.sha256
```

3. Extract the package.

```
tar -xzf tidb-v3.0-linux-amd64.tar.gz &&
cd tidb-v3.0-linux-amd64
```

3.2.2.1.5 Multiple nodes cluster deployment for test

If you want to test TiDB but have a limited number of nodes, you can use one PD instance to test the entire cluster.

Assuming that you have four nodes, you can deploy 1 PD instance, 3 TiKV instances, and 1 TiDB instance. See the following table for details:

Name	Host IP	Services
Node1	192.168.199.113	PD1, TiDB
Node2	192.168.199.114	TiKV1
Node3	192.168.199.115	TiKV2
Node4	192.168.199.116	TiKV3

Follow the steps below to start PD, TiKV and TiDB:

1. Start PD on Node1.

```
./bin/pd-server --name=pd1 \
                --data-dir=pd \
                --client-urls="http://192.168.199.113:2379" \
                --peer-urls="http://192.168.199.113:2380" \
                --initial-cluster="pd1=http://192.168.199.113:2380" \
                --log-file=pd.log &
```

2. Start TiKV on Node2, Node3 and Node4.

```
./bin/tikv-server --pd="192.168.199.113:2379" \
                  --addr="192.168.199.114:20160" \
                  --data-dir=tikv \
                  --log-file=tikv.log &

./bin/tikv-server --pd="192.168.199.113:2379" \
                  --addr="192.168.199.115:20160" \
                  --data-dir=tikv \
                  --log-file=tikv.log &

./bin/tikv-server --pd="192.168.199.113:2379" \
                  --addr="192.168.199.116:20160" \
                  --data-dir=tikv \
                  --log-file=tikv.log &
```

3. Start TiDB on Node1.

```
./bin/tidb-server --store=tikv \
                  --path="192.168.199.113:2379" \
                  --log-file=tidb.log
```

4. Use the MySQL client to connect to TiDB.

```
mysql -h 192.168.199.113 -P 4000 -u root -D test
```

3.2.2.2 Production Deployment from Binary Tarball

This guide provides installation instructions from a binary tarball on Linux. A complete TiDB cluster contains PD, TiKV, and TiDB. To start the database service, follow the order of PD -> TiKV -> TiDB. To stop the database service, follow the order of stopping TiDB -> TiKV -> PD.

See also local deployment and testing environment deployment.

3.2.2.2.1 Prepare

Before you start, see TiDB architecture and Software and Hardware Recommendations. Make sure the following requirements are satisfied:

Operating system

For the operating system, it is recommended to use RHEL/CentOS 7.3 or higher. The following additional requirements are recommended:

Configuration	Description
Supported Platform	RHEL/CentOS 7.3+ (more details)
File System	ext4 is recommended
Swap Space	Should be disabled
Disk Block Size	Set the system disk `Block` size to `4096`

Network and firewall

Configuration	Description
Firewall / Port	Check whether the ports required by TiDB are accessible between the nodes

Operating system parameters

Configuration	Description
Nice Limits	For system users, set the default value of `nice` in TiDB to `0`

Configuration	Description
min_free_kbytes	The setting for `vm.min_free_kbytes` in `sysctl.conf` needs to be high enough
User Open Files Limit	For database administrators, set the number of TiDB open files to `1000000`
System Open File Limits	Set the number of system open files to `1000000`
User Process Limits	For TiDB users, set the `nproc` value to `4096` in `limits.conf`
Address Space Limits	For TiDB users, set the space to `unlimited` in `limits.conf`
File Size Limits	For TiDB users, set the `fsize` value to `unlimited` in `limits.conf`
Disk Readahead	Set the value of the `readahead` data disk to `4096` at a minimum
NTP service	Configure the NTP time synchronization service for each node
SELinux	Turn off the SELinux service for each node
CPU Frequency Scaling	It is recommended to turn on CPU overclocking
Transparent Hugepages	For Red Hat 7+ and CentOS 7+ systems, it is required to set the Transparent Hugepages to `always`

Configuration	Description
I/O Scheduler	Set the I/O Scheduler of data disks to the `deadline` mode
vm.swappiness	Set `vm.swappiness = 0` in `sysctl.conf`
net.core.somaxconn	Set `net.core.somaxconn = 32768` in `sysctl.conf`
net.ipv4.tcp_syncookies	Set `net.ipv4.tcp_syncookies = 0` in `sysctl.conf`

Database running user settings

Configuration	Description
LANG environment	Set `LANG = en_US.UTF8`
TZ time zone	Set the TZ time zone of all nodes to the same value

3.2.2.2.2 TiDB components and default ports

Before you deploy a TiDB cluster, see the required components and optional components.

TiDB database components (required)

See the following table for the default ports for the TiDB components:

Component	Default Port	Protocol	Description
ssh	22	TCP	the sshd service
TiDB	4000	TCP	the communication port for the application and DBA tools
TiDB	10080	TCP	the communication port to report TiDB status
TiKV	20160	TCP	the TiKV communication port
PD	2379	TCP	the communication port between TiDB and PD
PD	2380	TCP	the inter-node communication port within the PD cluster

TiDB database components (optional)

See the following table for the default ports for the optional TiDB components:

Component	Default Port	Protocol	Description
Prometheus	9090	TCP	the communication port for the Prometheus service

Component	Default Port	Protocol	Description
Pushgateway	9091	TCP	the aggregation and report port for TiDB, TiKV, and PD monitor
Node_exporter	9100	TCP	the communication port to report the system information of every TiDB cluster node
Grafana	3000	TCP	the port for the external Web monitoring service and client (Browser) access
alertmanager	9093	TCP	the port for the alert service

3.2.2.2.3 Create a database running user account

1. Log in to the machine using the `root` user account and create a database running user account (`tidb`) using the following command:

```
useradd tidb -m
```

2. Switch the user from `root` to `tidb` by using the following command. You can use this `tidb` user account to deploy your TiDB cluster.

```
su - tidb
```

3.2.2.2.4 Download the official binary package

```
#### Download the package.
$ wget https://download.pingcap.org/tidb-{version}-linux-amd64.tar.gz
$ wget https://download.pingcap.org/tidb-{version}-linux-amd64.sha256

#### Check the file integrity. If the result is OK, the file is correct.
$ sha256sum -c tidb-{version}-linux-amd64.sha256

#### Extract the package.
$ tar -xzf tidb-{version}-linux-amd64.tar.gz
$ cd tidb-{version}-linux-amd64
```

Note:

`{version}` indicates the version number of TiDB. For example, if `{version}` is `v3.0.0`, the package download link is `https://download.pingcap.org/tidb-v3.0.0-linux-amd64.tar.gz`.

3.2.2.2.5 Multiple nodes cluster deployment

For the production environment, multiple nodes cluster deployment is recommended. Before you begin, see Software and Hardware Recommendations.

Assuming that you have six nodes, you can deploy 3 PD instances, 3 TiKV instances, and 1 TiDB instance. See the following table for details:

Name	Host IP	Services
Node1	192.168.199.113	PD1, TiDB
Node2	192.168.199.114	PD2
Node3	192.168.199.115	PD3
Node4	192.168.199.116	TiKV1
Node5	192.168.199.117	TiKV2
Node6	192.168.199.118	TiKV3

Follow the steps below to start PD, TiKV, and TiDB:

1. Start PD on Node1, Node2, and Node3 in sequence.

```
./bin/pd-server --name=pd1 \
                --data-dir=pd \
                --client-urls="http://192.168.199.113:2379" \
                --peer-urls="http://192.168.199.113:2380" \
                --initial-cluster="pd1=http://192.168.199.113:2380,pd2=
                   ↪ http://192.168.199.114:2380,pd3=http
                   ↪ ://192.168.199.115:2380" \
                -L "info" \
                --log-file=pd.log &

./bin/pd-server --name=pd2 \
                --data-dir=pd \
                --client-urls="http://192.168.199.114:2379" \
                --peer-urls="http://192.168.199.114:2380" \
                --initial-cluster="pd1=http://192.168.199.113:2380,pd2=
                   ↪ http://192.168.199.114:2380,pd3=http
```

```
                ↪ ://192.168.199.115:2380" \
              -L "info" \
              --log-file=pd.log &

./bin/pd-server --name=pd3 \
              --data-dir=pd \
              --client-urls="http://192.168.199.115:2379" \
              --peer-urls="http://192.168.199.115:2380" \
              --initial-cluster="pd1=http://192.168.199.113:2380,pd2=
                ↪ http://192.168.199.114:2380,pd3=http
                ↪ ://192.168.199.115:2380" \
              -L "info" \
              --log-file=pd.log &
```

2. Start TiKV on Node4, Node5 and Node6.

```
./bin/tikv-server --pd="
  ↪ 192.168.199.113:2379,192.168.199.114:2379,192.168.199.115:2379" \
              --addr="192.168.199.116:20160" \
              --status-addr="192.168.199.116:20180" \
              --data-dir=tikv \
              --log-file=tikv.log &

./bin/tikv-server --pd="
  ↪ 192.168.199.113:2379,192.168.199.114:2379,192.168.199.115:2379" \
              --addr="192.168.199.117:20160" \
              --status-addr="192.168.199.117:20180" \
              --data-dir=tikv \
              --log-file=tikv.log &

./bin/tikv-server --pd="
  ↪ 192.168.199.113:2379,192.168.199.114:2379,192.168.199.115:2379" \
              --addr="192.168.199.118:20160" \
              --status-addr="192.168.199.118:20180" \
              --data-dir=tikv \
              --log-file=tikv.log &
```

3. Start TiDB on Node1.

```
./bin/tidb-server --store=tikv \
              --path="
                ↪ 192.168.199.113:2379,192.168.199.114:2379,192.168.199.115:237
                ↪ " \
              --log-file=tidb.log &
```

4. Use the MySQL client to connect to TiDB.

```
mysql -h 192.168.199.113 -P 4000 -u root -D test
```

Note:

- If you start TiKV or deploy PD in the production environment, it is highly recommended to specify the path for the configuration file using the `--config` parameter. If the parameter is not set, TiKV or PD does not read the configuration file.
- To tune TiKV, see Performance Tuning for TiKV.
- If you use `nohup` to start the cluster in the production environment, write the startup commands in a script and then run the script. If not, the `nohup` process might abort because it receives exceptions when the Shell command exits. For more information, see The TiDB/TiKV/PD process aborts unexpectedly.

For the deployment and use of TiDB monitoring services, see Monitor a TiDB Cluster.

3.2.3 Orchestrated Deployment

3.2.3.1 Deploy TiDB Using TiDB Ansible

This guide describes how to deploy a TiDB cluster using TiDB Ansible. For the production environment, it is recommended to deploy TiDB using TiDB Ansible.

3.2.3.1.1 Overview

Ansible is an IT automation tool that can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.

TiDB Ansible is a TiDB cluster deployment tool developed by PingCAP, based on Ansible playbook. TiDB Ansible enables you to quickly deploy a new TiDB cluster which includes PD, TiDB, TiKV, and the cluster monitoring modules.

You can use the TiDB Ansible configuration file to set up the cluster topology and complete all the following operation tasks:

- Initialize operating system parameters
- Deploy the whole TiDB cluster
- Start the TiDB cluster
- Stop the TiDB cluster

- Modify component configuration
- Scale the TiDB cluster
- Upgrade the component version
- Enable the cluster binlog
- Clean up data of the TiDB cluster
- Destroy the TiDB cluster

3.2.3.1.2 Prepare

Before you start, make sure you have:

1. Several target machines that meet the following requirements:

    - 4 or more machines

      A standard TiDB cluster contains 6 machines. You can use 4 machines for testing. For more details, see Software and Hardware Recommendations.
    - CentOS 7.3 (64 bit) or later, x86_64 architecture (AMD64)
    - Network between machines

> **Note:**
> When you deploy TiDB using TiDB Ansible, **use SSD disks for the data directory of TiKV and PD nodes**. Otherwise, it cannot pass the check. If you only want to try TiDB out and explore the features, it is recommended to deploy TiDB using Docker Compose on a single machine.

2. A Control Machine that meets the following requirements:

> **Note:**
> The Control Machine can be one of the target machines.

    - CentOS 7.3 (64 bit) or later with Python 2.7 installed
    - Access to the Internet

3.2.3.1.3 Step 1: Install system dependencies on the Control Machine

Log in to the Control Machine using the `root` user account, and run the corresponding command according to your operating system.

- If you use a Control Machine installed with CentOS 7, run the following command:

```
yum -y install epel-release git curl sshpass &&
yum -y install python2-pip
```

- If you use a Control Machine installed with Ubuntu, run the following command:

```
apt-get -y install git curl sshpass python-pip
```

3.2.3.1.4 Step 2: Create the `tidb` user on the Control Machine and generate the SSH key

Make sure you have logged in to the Control Machine using the `root` user account, and then run the following command.

1. Create the `tidb` user.

```
useradd -m -d /home/tidb tidb
```

2. Set a password for the `tidb` user account.

```
passwd tidb
```

3. Configure sudo without password for the `tidb` user account by adding `tidb ALL=(ALL ↪ )NOPASSWD: ALL` to the end of the sudo file:

```
visudo
```

```
tidb ALL=(ALL) NOPASSWD: ALL
```

4. Generate the SSH key.

   Execute the `su` command to switch the user from `root` to `tidb`.

```
su - tidb
```

   Create the SSH key for the `tidb` user account and hit the Enter key when `Enter ↪ passphrase` is prompted. After successful execution, the SSH private key file is `/home/tidb/.ssh/id_rsa`, and the SSH public key file is `/home/tidb/.ssh/id_rsa. ↪ pub`.

```
ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tidb/.ssh/id_rsa):
Created directory '/home/tidb/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

```
Your identification has been saved in /home/tidb/.ssh/id_rsa.
Your public key has been saved in /home/tidb/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:eIBykszR1KyECA/h0d7PRKz4fhAeli7IrVphhte7/So tidb@172.16.10.49
The key's randomart image is:
+---[RSA 2048]----+
|=+o+.o.          |
|o=o+o.oo         |
| .O.=.=          |
| . B.B +         |
|o B * B S        |
| * + * +         |
|  o + .          |
| o  E+ .         |
|o   ..+o.        |
+----[SHA256]-----+
```

3.2.3.1.5 Step 3: Download TiDB Ansible to the Control Machine

Log in to the Control Machine using the `tidb` user account and enter the `/home/tidb` directory. Run the following command to download the corresponding TAG version of TiDB Ansible 3.0 from the TiDB Ansible project. The default folder name is `tidb-ansible`.

```
git clone -b $tag https://github.com/pingcap/tidb-ansible.git
```

Note:

- Replace `$tag` with the value of the chosen TAG version. For example, `v3.0.2`.
- To deploy and upgrade TiDB clusters, use the corresponding version of `tidb-ansible`. If you only modify the version in the `inventory.ini` file, errors might occur.
- It is required to download `tidb-ansible` to the `/home/tidb` directory using the `tidb` user account. If you download it to the `/root` directory, a privilege issue occurs.

If you have questions regarding which version to use, email to info@pingcap.com for more information or file an issue.

3.2.3.1.6 Step 4: Install TiDB Ansible and its dependencies on the Control Machine

Make sure you have logged in to the Control Machine using the `tidb` user account.

It is required to use `pip` to install Ansible and its dependencies, otherwise a compatibility issue occurs. Currently, the release-2.0, release-2.1, and master branches of TiDB Ansible are compatible with Ansible 2.4 ~ 2.7.11 (2.4 Ansible 2.7.11).

1. Install TiDB Ansible and the dependencies on the Control Machine:

```
cd /home/tidb/tidb-ansible &&
sudo pip install -r ./requirements.txt
```

   The version information of Ansible and dependencies is in the `tidb-ansible/requirements.txt` file.

2. View the version of TiDB Ansible:

```
ansible --version
```

```
ansible 2.7.11
```

3.2.3.1.7 Step 5: Configure the SSH mutual trust and sudo rules on the Control Machine

Make sure you have logged in to the Control Machine using the `tidb` user account.

1. Add the IPs of your target machines to the `[servers]` section of the `hosts.ini` file.

```
cd /home/tidb/tidb-ansible &&
vi hosts.ini
```

```
[servers]
172.16.10.1
172.16.10.2
172.16.10.3
172.16.10.4
172.16.10.5
172.16.10.6

[all:vars]
username = tidb
ntp_server = pool.ntp.org
```

2. Run the following command and input the `root` user account password of your target machines.

```
ansible-playbook -i hosts.ini create_users.yml -u root -k
```

This step creates the `tidb` user account on the target machines, configures the sudo rules and the SSH mutual trust between the Control Machine and the target machines.

To configure the SSH mutual trust and sudo without password manually, see How to manually configure the SSH mutual trust and sudo without password.

3.2.3.1.8 Step 6: Install the NTP service on the target machines

> **Note:**
>
> If the time and time zone of all your target machines are same, the NTP service is on and is normally synchronizing time, you can ignore this step. See How to check whether the NTP service is normal.

Make sure you have logged in to the Control Machine using the `tidb` user account, run the following command:

```
cd /home/tidb/tidb-ansible &&
ansible-playbook -i hosts.ini deploy_ntp.yml -u tidb -b
```

The NTP service is installed and started using the software repository that comes with the system on the target machines. The default NTP server list in the installation package is used. The related `server` parameter is in the `/etc/ntp.conf` configuration file.

To make the NTP service start synchronizing as soon as possible, the system executes the `ntpdate` command to set the local date and time by polling `ntp_server` in the `hosts.ini` file. The default server is `pool.ntp.org`, and you can also replace it with your NTP server.

3.2.3.1.9 Step 7: Configure the CPUfreq governor mode on the target machine

For details about CPUfreq, see the CPUfreq Governor documentation.

Set the CPUfreq governor mode to `performance` to make full use of CPU performance.

Check the governor modes supported by the system

You can run the `cpupower frequency-info --governors` command to check the governor modes which the system supports:

```
cpupower frequency-info --governors
```

```
analyzing CPU 0:
  available cpufreq governors: performance powersave
```

Taking the above code for example, the system supports the `performance` and `powersave` modes.

> **Note:**
>
> As the following shows, if it returns `Not Available`, it means that the current system does not support CPUfreq configuration and you can skip this step.

```
cpupower frequency-info --governors
```

```
analyzing CPU 0:
  available cpufreq governors: Not Available
```

Check the current governor mode

You can run the `cpupower frequency-info --policy` command to check the current CPUfreq governor mode:

```
cpupower frequency-info --policy
```

```
analyzing CPU 0:
  current policy: frequency should be within 1.20 GHz and 3.20 GHz.
                  The governor "powersave" may decide which speed to use
                  within this range.
```

As the above code shows, the current mode is `powersave` in this example.

Change the governor mode

You can use either of the following two methods to change the governor mode. In the above example, the current governor mode is `powersave` and the following commands change it to `performance`.

- Use the `cpupower frequency-set --governor` command to change the current mode:

  ```
  cpupower frequency-set --governor performance
  ```

- Run the following command to set the mode on the target machine in batches:

  ```
  ansible -i hosts.ini all -m shell -a "cpupower frequency-set --governor
      ↪ performance" -u tidb -b
  ```

3.2.3.1.10 Step 8: Mount the data disk ext4 filesystem with options on the target machines

Log in to the target machines using the `root` user account.

Format your data disks to the ext4 filesystem and add the `nodelalloc` and `noatime` mount options to the filesystem. It is required to add the `nodelalloc` option, or else the Ansible deployment cannot pass the test. The `noatime` option is optional.

> **Note:**
>
> If your data disks have been formatted to ext4 and have added the mount options, you can uninstall it by running the `umount /dev/nvme0n1p1` command, follow the steps starting from editing the `/etc/fstab` file, and add the options again to the filesystem.

Take the `/dev/nvme0n1` data disk as an example:

1. View the data disk.

```
fdisk -l
```

```
Disk /dev/nvme0n1: 1000 GB
```

2. Create the partitioned table.

```
parted -s -a optimal /dev/nvme0n1 mklabel gpt -- mkpart primary ext4 1
    ↪ -1
```

> **Note:**
>
> Use the `lsblk` command to view the device number of the partition: for a nvme disk, the generated device number is usually `nvme0n1p1`; for a regular disk (for example, `/dev/sdb`), the generated device number is usually `sdb1`.

3. Format the data disk to the ext4 filesystem.

```
mkfs.ext4 /dev/nvme0n1p1
```

4. View the partition UUID of the data disk.

   In this example, the UUID of `nvme0n1p1` is `c51eb23b-195c-4061-92a9-3fad812cc12f` ↪ .

```
lsblk -f
```

```
NAME   FSTYPE LABEL UUID                                 MOUNTPOINT
sda
 ─sda1 ext4        237b634b-a565-477b-8371-6dff0c41f5ab /boot
 ─sda2 swap        f414c5c0-f823-4bb1-8fdf-e531173a72ed
 ─sda3 ext4        547909c1-398d-4696-94c6-03e43e317b60 /
sr0
nvme0n1
 ─nvme0n1p1 ext4       c51eb23b-195c-4061-92a9-3fad812cc12f
```

5. Edit the `/etc/fstab` file and add the mount options.

```
vi /etc/fstab
```

```
UUID=c51eb23b-195c-4061-92a9-3fad812cc12f /data1 ext4 defaults,
   ↪ nodelalloc,noatime 0 2
```

6. Mount the data disk.

```
mkdir /data1 &&
mount -a
```

7. Check using the following command.

```
mount -t ext4
```

```
/dev/nvme0n1p1 on /data1 type ext4 (rw,noatime,nodelalloc,data=ordered)
```

If the filesystem is ext4 and `nodelalloc` is included in the mount options, you have successfully mount the data disk ext4 filesystem with options on the target machines.

3.2.3.1.11 Step 9: Edit the `inventory.ini` file to orchestrate the TiDB cluster

Log in to the Control Machine using the `tidb` user account, and edit the `tidb-ansible/inventory.ini` file to orchestrate the TiDB cluster. The standard TiDB cluster contains 6 machines: 2 TiDB instances, 3 PD instances, and 3 TiKV instances.

- Deploy at least 3 instances for TiKV.
- Do not deploy TiKV together with TiDB or PD on the same machine.
- Use the first TiDB machine as the monitoring machine.

Note:

It is required to use the internal IP address to deploy. If the SSH port of the target machines is not the default 22 port, you need to add the `ansible_port` variable. For example, `TiDB1 ansible_host=172.16.10.1 ansible_port =5555`.

You can choose one of the following two types of cluster topology according to your scenario:

- The cluster topology of a single TiKV instance on each TiKV node

  In most cases, it is recommended to deploy one TiKV instance on each TiKV node for better performance. However, if the CPU and memory of your TiKV machines are much better than the required in Hardware and Software Requirements, and you have more than two disks in one node or the capacity of one SSD is larger than 2 TB, you can deploy no more than 2 TiKV instances on a single TiKV node.

- The cluster topology of multiple TiKV instances on each TiKV node

Option 1: Use the cluster topology of a single TiKV instance on each TiKV node

Name	Host IP	Services
node1	172.16.10.1	PD1, TiDB1
node2	172.16.10.2	PD2, TiDB2
node3	172.16.10.3	PD3
node4	172.16.10.4	TiKV1
node5	172.16.10.5	TiKV2
node6	172.16.10.6	TiKV3

```
[tidb_servers]
172.16.10.1
172.16.10.2

[pd_servers]
172.16.10.1
172.16.10.2
172.16.10.3

[tikv_servers]
172.16.10.4
```

```
172.16.10.5
172.16.10.6

[monitoring_servers]
172.16.10.1

[grafana_servers]
172.16.10.1

[monitored_servers]
172.16.10.1
172.16.10.2
172.16.10.3
172.16.10.4
172.16.10.5
172.16.10.6
```

Option 2: Use the cluster topology of multiple TiKV instances on each TiKV node

Take two TiKV instances on each TiKV node as an example:

Name	Host IP	Services
node1	172.16.10.1	PD1, TiDB1
node2	172.16.10.2	PD2, TiDB2
node3	172.16.10.3	PD3
node4	172.16.10.4	TiKV1-1, TiKV1-2
node5	172.16.10.5	TiKV2-1, TiKV2-2
node6	172.16.10.6	TiKV3-1, TiKV3-2

```
[tidb_servers]
172.16.10.1
172.16.10.2

[pd_servers]
172.16.10.1
172.16.10.2
172.16.10.3

#### Note: To use labels in TiKV, you must also configure location_labels
   ↪ for PD at the same time.
[tikv_servers]
TiKV1-1 ansible_host=172.16.10.4 deploy_dir=/data1/deploy tikv_port=20171
   ↪ labels="host=tikv1"
TiKV1-2 ansible_host=172.16.10.4 deploy_dir=/data2/deploy tikv_port=20172
```

```
   ↪ labels="host=tikv1"
TiKV2-1 ansible_host=172.16.10.5 deploy_dir=/data1/deploy tikv_port=20171
   ↪ labels="host=tikv2"
TiKV2-2 ansible_host=172.16.10.5 deploy_dir=/data2/deploy tikv_port=20172
   ↪ labels="host=tikv2"
TiKV3-1 ansible_host=172.16.10.6 deploy_dir=/data1/deploy tikv_port=20171
   ↪ labels="host=tikv3"
TiKV3-2 ansible_host=172.16.10.6 deploy_dir=/data2/deploy tikv_port=20172
   ↪ labels="host=tikv3"

#### When you deploy a TiDB cluster of the 3.0 version, you must configure
   ↪ the TiKV status ports in the topology of multiple TiKV instances, as
   ↪ shown in the following example.
#### TiKV1-1 ansible_host=172.16.10.4 deploy_dir=/data1/deploy tikv_port
   ↪ =20171 tikv_status_port=20181 labels="host=tikv1"
#### TiKV1-2 ansible_host=172.16.10.4 deploy_dir=/data2/deploy tikv_port
   ↪ =20172 tikv_status_port=20182 labels="host=tikv1"
#### TiKV2-1 ansible_host=172.16.10.5 deploy_dir=/data1/deploy tikv_port
   ↪ =20171 tikv_status_port=20181 labels="host=tikv2"
#### TiKV2-2 ansible_host=172.16.10.5 deploy_dir=/data2/deploy tikv_port
   ↪ =20172 tikv_status_port=20182 labels="host=tikv2"
#### TiKV3-1 ansible_host=172.16.10.6 deploy_dir=/data1/deploy tikv_port
   ↪ =20171 tikv_status_port=20181 labels="host=tikv3"
#### TiKV3-2 ansible_host=172.16.10.6 deploy_dir=/data2/deploy tikv_port
   ↪ =20172 tikv_status_port=20182 labels="host=tikv3"

[monitoring_servers]
172.16.10.1

[grafana_servers]
172.16.10.1

[monitored_servers]
172.16.10.1
172.16.10.2
172.16.10.3
172.16.10.4
172.16.10.5
172.16.10.6

......

#### Note: For labels in TiKV to work, you must also configure
   ↪ location_labels for PD when deploying the cluster.
[pd_servers:vars]
```

```
location_labels = ["host"]
```

Edit the parameters in the service configuration file:

1. For the cluster topology of multiple TiKV instances on each TiKV node, you need to edit the `capacity` parameter under `block-cache-size` in `tidb-ansible/conf/tikv.yml`:

```
storage:
  block-cache:
    capacity: "1GB"
```

> **Note:**
>
> - The number of TiKV instances is the number of TiKV processes on each server.
> - Recommended configuration: `capacity` = MEM_TOTAL * 0.5 / the number of TiKV instances

2. For the cluster topology of multiple TiKV instances on each TiKV node, you need to edit the `high-concurrency`, `normal-concurrency` and `low-concurrency` parameters in the `tidb-ansible/conf/tikv.yml` file:

```
readpool:
coprocessor:
    # Notice: if CPU_NUM > 8, default thread pool size for coprocessors
    # will be set to CPU_NUM * 0.8.
    # high-concurrency: 8
    # normal-concurrency: 8
    # low-concurrency: 8
```

> **Note:**
>
> Recommended configuration: the number of TiKV instances * the parameter value = the number of CPU cores * 0.8.

3. If multiple TiKV instances are deployed on a same physical disk, edit the `capacity` parameter in `conf/tikv.yml`:

```
raftstore:
  capacity: 0
```

> **Note:**
>
> Recommended configuration: `capacity` = total disk capacity / the number of TiKV instances. For example, `capacity: "100GB"`.

3.2.3.1.12 Step 10: Edit variables in the `inventory.ini` file

This step describes how to edit the variable of deployment directory and other variables in the `inventory.ini` file.

Configure the deployment directory

Edit the `deploy_dir` variable to configure the deployment directory.

The global variable is set to `/home/tidb/deploy` by default, and it applies to all services. If the data disk is mounted on the `/data1` directory, you can set it to `/data1/deploy`. For example:

```
##### Global variables
[all:vars]
deploy_dir = /data1/deploy
```

> **Note:**
>
> To separately set the deployment directory for a service, you can configure the host variable while configuring the service host list in the `inventory.ini` file. It is required to add the first column alias, to avoid confusion in scenarios of mixed services deployment.

```
TiKV1-1 ansible_host=172.16.10.4 deploy_dir=/data1/deploy
```

Edit other variables (Optional)

To enable the following control variables, use the capitalized `True`. To disable the following control variables, use the capitalized `False`.

Variable Name	Description
cluster_name	the name of a cluster, adjustable

Variable Name	Description
tidb_version	show version of TiDB, configured by default in TiDB Ansible branches
process_supervision	the service supervision way of processes, systemd by default, supervise optional

Variable Name	Description
timezone	the global default time zone configured when a new TiDB cluster bootstrap is initialized; you can edit it later using the global `time_zone` system variable and the session `time_zone` system variable as described in Time Zone Support; the default value is `Asia/Shanghai` and see the list of time zones for more optional values

Variable Name	Description
enable_firewall	to enable the firewall, closed by default; to enable it, add the ports in network requirements to the allowlist
enable_ntpd	to monitor the NTP service of the managed node, True by default; do not close it
set_hostname	to edit the hostname of the managed node based on the IP, False by default

Variable Name	Description
enable_binlog	whether to deploy Pump and enable the binlog, False by default, dependent on the Kafka cluster; see the `zookeeper_addrs` variable
zookeeper_addrs	the zookeeper address of the binlog Kafka cluster

Variable Name	Description
deploy_without_tidb	this Key-Value mode, deploy only PD, TiKV and the monitoring service, not TiDB; set the IP of the tidb_servers host group to null in the `inventory.ini` file
alertmanager_target	optional. If you have deployed `alertmanager` separately, you can configure this variable using the `alertmanager_host:alertmanager_port` format

Variable Name	Description
grafana_admin_user	the username of Grafana administrator; default `admin`
grafana_admin_password	the password of Grafana administrator account; default `admin`; used to import Dashboard and create the API key using TiDB Ansible; update this variable if you have modified it through Grafana web

Variable Name	Description
collect_log_recent_hours	to collect the log of recent hours; default the recent 2 hours
enable_bandwidth_limit	to set a band-width limit when pulling the diagnostic data from the target machines to the Control Machine; used together with the `collect_bandwidth_limit` variable

Variable Name	Description
collect_bandwidth_limit	the limited bandwidth when pulling the diagnostic data from the target machines to the Control Machine; unit: Kbit/s; default 10000, indicating 10Mb/s; for the cluster topology of multiple TiKV instances on each TiKV node, you need to divide the number of the TiKV instances on each TiKV node

Variable Name	Description
prometheus_storage_retention	the retention time of the monitoring data of Prometheus (30 days by default); this is a new configuration in the `group_vars/monitoring_servers.yml` file in 2.1.7, 3.0 and the later tidb-ansible versions

3.2.3.1.13 Step 11: Deploy the TiDB cluster

When `ansible-playbook` runs Playbook, the default concurrent number is 5. If many deployment target machines are deployed, you can add the `-f` parameter to specify the concurrency, such as `ansible-playbook deploy.yml -f 10`.

The following example uses `tidb` as the user who runs the service.

1. Edit the `tidb-ansible/inventory.ini` file to make sure `ansible_user = tidb`.

```
## Connection
# ssh via normal user
ansible_user = tidb
```

Note:

Do not configure `ansible_user` to `root`, because `tidb-ansible` limits the user that runs the service to the normal user.

Run the following command and if all servers return `tidb`, then the SSH mutual trust is successfully configured:

```
ansible -i inventory.ini all -m shell -a 'whoami'
```

Run the following command and if all servers return `root`, then sudo without password of the `tidb` user is successfully configured:

```
ansible -i inventory.ini all -m shell -a 'whoami' -b
```

2. Run the `local_prepare.yml` playbook and download TiDB binary to the Control Machine.

```
ansible-playbook local_prepare.yml
```

3. Initialize the system environment and modify the kernel parameters.

```
ansible-playbook bootstrap.yml
```

4. Deploy the TiDB cluster software.

```
ansible-playbook deploy.yml
```

Note:

You can use the `Report` button on the Grafana Dashboard to generate the PDF file. This function depends on the `fontconfig` package and English fonts. To use this function, log in to the `grafana_servers` machine and install it using the following command:

```
sudo yum install fontconfig open-sans-fonts
```

5. Start the TiDB cluster.

```
ansible-playbook start.yml
```

3.2.3.1.14 Test the TiDB cluster

Because TiDB is compatible with MySQL, you must use the MySQL client to connect to TiDB directly. It is recommended to configure load balancing to provide uniform SQL interface.

1. Connect to the TiDB cluster using the MySQL client.

```
mysql -u root -h 172.16.10.1 -P 4000
```

Note:
The default port of TiDB service is `4000`.

2. Access the monitoring platform using a web browser.

    - Address: http://172.16.10.1:3000
    - Default account and password: `admin`; `admin`

3.2.3.1.15 Deployment FAQs

This section lists the common questions about deploying TiDB using TiDB Ansible.

How to customize the port?

Edit the `inventory.ini` file and add the following host variable after the IP of the corresponding service:

Component	Variable Port	Default Port	Description
TiDB	tidb_port	4000	the communication port for the application and DBA tools
TiDB	tidb_status_port	10080	the communication port to report TiDB status
TiKV	tikv_port	20160	the TiKV communication port
TiKV	tikv_status_port	20180	the communication port to report the TiKV status
PD	pd_client_port	2379	the communication port between TiDB and PD
PD	pd_peer_port	2380	the inter-node communication port within the PD cluster
Pump	pump_port	8250	the pump communication port
Prometheus	prometheus_port	9090	the communication port for the Prometheus service
Pushgateway	pushgateway_port	9091	the aggregation and report port for TiDB, TiKV, and PD monitor
Node_exporter	node_exporter_port	9100	the communication port to report the system information of every TiDB cluster node

Component	Variable Port	Default Port	Description
Grafana	grafana_port	3000	the port for the external Web monitoring service and client (Browser) access
Kafka_exporter	kafka_exporter_port	9308	the communication port for Kafka_exporter, used to monitor the binlog Kafka cluster

How to customize the deployment directory?

Edit the `inventory.ini` file and add the following host variable after the IP of the corresponding service:

Component	Variable Directory	Default Directory	Description
Global	deploy_dir	/home/tidb/deploy	the deployment directory
TiDB	tidb_log_dir	{{ deploy_dir }}/log	the TiDB log directory
TiKV	tikv_log_dir	{{ deploy_dir }}/log	the TiKV log directory
TiKV	tikv_data_dir	{{ deploy_dir }}/data	the data directory

Component	Variable Directory	Default Directory	Description
TiKV	wal_dir	“”	the rocksdb write-ahead log directory, consistent with the TiKV data directory when the value is null
TiKV	raftdb_path	“”	the raftdb directory, being tikv_data_dir/raft when the value is null
PD	pd_log_dir	{{ deploy_dir }}/log	the PD log directory
PD	pd_data_dir	{{ deploy_dir }}/data.pd	the PD data directory
Pump	pump_log_dir	{{ deploy_dir }}/log	the Pump log directory

Component	Variable Directory	Default Directory	Description
Pump	pump_data_dir	{{ deploy_dir }}/data.pump	the Pump data directory
Prometheus	prometheus_log_dir	{{ deploy_dir }}/log	the Prometheus log directory
Prometheus	prometheus_data_dir	{{ deploy_dir }}/data.metrics	the Prometheus data directory
Pushgateway	pushgateway_log_dir	{{ deploy_dir }}/log	the pushgateway log directory
Node_exporter	node_exporter_log_dir	{{ deploy_dir }}/log	the node_exporter log directory
Grafana	grafana_log_dir	{{ deploy_dir }}/log	the Grafana log directory
Grafana	grafana_data_dir	{{ deploy_dir }}/data.grafana	the Grafana data directory

How to check whether the NTP service is normal?

1. Run the following command. If it returns `running`, then the NTP service is running:

```
sudo systemctl status ntpd.service
```

```
ntpd.service - Network Time Service
Loaded: loaded (/usr/lib/systemd/system/ntpd.service; disabled; vendor
   ↪ preset: disabled)
```

```
Active: active (running) since — 2017-12-18 13:13:19 CST; 3s ago
```

2. Run the ntpstat command. If it returns `synchronised to NTP server` (synchronizing with the NTP server), then the synchronization process is normal.

```
ntpstat
```

```
synchronised to NTP server (85.199.214.101) at stratum 2
time correct to within 91 ms
polling server every 1024 s
```

Note:

For the Ubuntu system, you need to install the `ntpstat` package.

- The following condition indicates the NTP service is not synchronizing normally:

```
ntpstat
```

```
unsynchronised
```

- The following condition indicates the NTP service is not running normally:

```
ntpstat
```

```
Unable to talk to NTP daemon. Is it running?
```

- To make the NTP service start synchronizing as soon as possible, run the following command. You can replace `pool.ntp.org` with other NTP servers.

```
sudo systemctl stop ntpd.service &&
sudo ntpdate pool.ntp.org &&
sudo systemctl start ntpd.service
```

- To install the NTP service manually on the CentOS 7 system, run the following command:

```
sudo yum install ntp ntpdate &&
sudo systemctl start ntpd.service &&
sudo systemctl enable ntpd.service
```

How to modify the supervision method of a process from `supervise` to `systemd`?

Run the following command:

```
process supervision, [systemd, supervise]
```

```
process_supervision = systemd
```

For versions earlier than TiDB 1.0.4, the process supervision method of TiDB Ansible is `supervise` by default. The previously installed cluster can remain the same. If you need to change the supervision method to `systemd`, stop the cluster and run the following command:

```
ansible-playbook stop.yml
ansible-playbook deploy.yml -D &&
ansible-playbook start.yml
```

How to manually configure the SSH mutual trust and sudo without password?

1. Log in to the deployment target machine respectively using the `root` user account, create the `tidb` user and set the login password.

    ```
    useradd tidb &&
    passwd tidb
    ```

2. To configure sudo without password, run the following command, and add `tidb ALL =(ALL)NOPASSWD: ALL` to the end of the file:

    ```
    visudo
    ```

    ```
    tidb ALL=(ALL) NOPASSWD: ALL
    ```

3. Use the `tidb` user to log in to the Control Machine, and run the following command. Replace `172.16.10.61` with the IP of your deployment target machine, and enter the `tidb` user password of the deployment target machine as prompted. Successful execution indicates that SSH mutual trust is already created. This applies to other machines as well.

    ```
    ssh-copy-id -i ~/.ssh/id_rsa.pub 172.16.10.61
    ```

4. Log in to the Control Machine using the `tidb` user account, and log in to the IP of the target machine using SSH. If you do not need to enter the password and can successfully log in, then the SSH mutual trust is successfully configured.

    ```
    ssh 172.16.10.61
    ```

    ```
    [tidb@172.16.10.61 ~]$
    ```

5. After you login to the deployment target machine using the `tidb` user, run the following command. If you do not need to enter the password and can switch to the `root` user, then sudo without password of the `tidb` user is successfully configured.

```
sudo -su root
```

```
[root@172.16.10.61 tidb]#
```

Error: You need to install jmespath prior to running json_query filter

1. See Install TiDB Ansible and its dependencies on the Control Machine and use `pip` to install TiDB Ansible and the corresponding dependencies in the Control Machine. The `jmespath` dependent package is installed by default.

2. Run the following command to check whether `jmespath` is successfully installed:

```
pip show jmespath
```

```
Name: jmespath
Version: 0.9.0
```

3. Enter `import jmespath` in the Python interactive window of the Control Machine.

    - If no error displays, the dependency is successfully installed.
    - If the `ImportError: No module named jmespath` error displays, the Python `jmespath` module is not successfully installed.

```
python
```

```
Python 2.7.5 (default, Nov 6 2016, 00:28:07)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```

```
import jmespath
```

The `zk: node does not exist` error when starting Pump/Drainer

Check whether the `zookeeper_addrs` configuration in `inventory.ini` is the same with the configuration in the Kafka cluster, and whether the namespace is filled in. The description about namespace configuration is as follows:

```
#### ZooKeeper connection string (see ZooKeeper docs for details).
#### ZooKeeper address of the Kafka cluster. Example:
#### zookeeper_addrs =
   ↪ "192.168.0.11:2181,192.168.0.12:2181,192.168.0.13:2181"
#### You can also append an optional chroot string to the URLs to specify
   ↪ the root directory for all Kafka znodes. Example:
#### zookeeper_addrs =
   ↪ "192.168.0.11:2181,192.168.0.12:2181,192.168.0.13:2181/kafka/123"
```

3.2.3.2 Deploy TiDB Offline Using TiDB Ansible

This guide describes how to deploy a TiDB cluster offline using TiDB Ansible.

3.2.3.2.1 Prepare

Before you start, make sure that you have:

1. A download machine
    - The machine must have access to the Internet in order to download TiDB Ansible, TiDB and related packages.
    - For Linux operating system, it is recommended to install CentOS 7.3 or later.
2. Several target machines and one Control Machine
    - For system requirements and configuration, see Prepare the environment.
    - It is acceptable without access to the Internet.

3.2.3.2.2 Step 1: Install system dependencies on the Control Machine

Take the following steps to install system dependencies on the Control Machine installed with the CentOS 7 system.

1. Download the `pip` offline installation package on the download machine and then upload it to the Control Machine.

> **Note:**
> This offline installation package includes `pip` and `sshpass`, and only supports the CentOS 7 system.

2. Install system dependencies on the Control Machine.

```
tar -xzvf ansible-system-rpms.el7.tar.gz &&
cd ansible-system-rpms.el7 &&
chmod u+x install_ansible_system_rpms.sh &&
./install_ansible_system_rpms.sh
```

3. After the installation is finished, you can use `pip -V` to check whether it is successfully installed.

```
pip -V
```

```
pip 8.1.2 from /usr/lib/python2.7/site-packages (python 2.7)
```

Note:

If `pip` is already installed to your system, make sure that the version is 8.1.2 or later. Otherwise, compatibility error occurs when you install TiDB Ansible and its dependencies offline.

3.2.3.2.3 Step 2: Create the `tidb` user on the Control Machine and generate the SSH key

See Create the `tidb` user on the Control Machine and generate the SSH key.

3.2.3.2.4 Step 3: Install TiDB Ansible and its dependencies offline on the Control Machine

Currently, all the versions of TiDB Ansible from 2.4 to 2.7.11 are supported. The versions of TiDB Ansible and the related dependencies are listed in the `tidb-ansible/requirements.txt` file. The following installation steps take Ansible 2.5 as an example.

1. Download Ansible 2.5 offline installation package on the download machine and then upload it to the Control Machine.
2. Install TiDB Ansible and its dependencies offline.

```
tar -xzvf ansible-2.5.0-pip.tar.gz &&
cd ansible-2.5.0-pip/ &&
chmod u+x install_ansible.sh &&
./install_ansible.sh
```

3. View the version of TiDB Ansible.

   After TiDB Ansible is installed, you can view the version using `ansible --version`.

```
ansible --version
```

```
ansible 2.5.0
```

3.2.3.2.5 Step 4: Download TiDB Ansible and TiDB packages on the download machine

1. Install TiDB Ansible on the download machine.

   Use the following method to install Ansible online on the download machine installed with the CentOS 7 system. After TiDB Ansible is installed, you can view the version using `ansible --version`.

```
yum install epel-release &&
yum install ansible curl &&
ansible --version
```

```
ansible 2.5.0
```

> **Note:**
> Make sure that the version of Ansible is 2.5, otherwise a compatibility issue occurs.

2. Download TiDB Ansible.

   Use the following command to download TiDB 3.0 version with the corresponding tag from the TiDB Ansible project GitHub repo. The default folder name is `tidb-ansible`.

```
git clone -b $tag https://github.com/pingcap/tidb-ansible.git
```

> **Note:**
>
> - Replace `$tag` with the value of the selected tag version, such as `v3.0.2`.
> - It is required to use the corresponding tidb-ansible version when you deploy and upgrade the TiDB cluster. If you deploy TiDB using a mismatched version of tidb-ansible (such as using tidb-ansible v2.1.4 to deploy TiDB v2.1.6), an error might occur.

3. Run the `local_prepare.yml` playbook, and download TiDB binary online to the download machine.

```
cd tidb-ansible &&
ansible-playbook local_prepare.yml
```

4. After running the above command, copy the `tidb-ansible` folder to the `/home/tidb` directory of the Control Machine. The ownership authority of the file must be the `tidb` user.

3.2.3.2.6 Step 5: Configure the SSH mutual trust and sudo rules on the Control Machine

See Configure the SSH mutual trust and sudo rules on the Control Machine.

3.2.3.2.7 Step 6: Install the NTP service on the target machines

See Install the NTP service on the target machines.

> **Note:**
>
> If the time and time zone of all your target machines are same, the NTP service is on and is normally synchronizing time, you can skip this step. See How to check whether the NTP service is normal.

3.2.3.2.8 Step 7: Configure the CPUfreq governor mode on the target machine

See Configure the CPUfreq governor mode on the target machine.

3.2.3.2.9 Step 8: Mount the data disk ext4 filesystem with options on the target machines

See Mount the data disk ext4 filesystem with options on the target machines.

3.2.3.2.10 Step 9: Edit the `inventory.ini` file to orchestrate the TiDB cluster

See Edit the `inventory.ini` file to orchestrate the TiDB cluster.

3.2.3.2.11 Step 10: Deploy the TiDB cluster

1. You do not need to run the playbook in `ansible-playbook local_prepare.yml`.
2. See Deploy the TiDB cluster.

3.2.3.2.12 Test the TiDB cluster

See Test the TiDB cluster.

3.2.3.3 Deploy TiDB Using Docker

> **Warning:**
>
> The Docker deployment method provided in this document is no longer maintained. If you want to test TiDB, it is recommended to refer to Quick Start Guide for the TiDB Database Platform for deployment. For **production environment**, **do not use** Docker for deployment, but deploy TiDB with TiDB Ansible or TiDB Operator in Kubernetes.

This document shows you how to manually deploy a multi-node TiDB cluster on multiple machines using Docker.

To learn more, see TiDB architecture and Software and Hardware Recommendations.

3.2.3.3.1 Preparation

Before you start, make sure that you have:

- Installed the latest version of Docker
- Pulled the latest images of TiDB, TiKV and PD from Docker Hub. If not, pull the images using the following commands:

```
docker pull pingcap/tidb:latest
```

```
docker pull pingcap/tikv:latest
```

```
docker pull pingcap/pd:latest
```

3.2.3.3.2 Multi nodes deployment

Assume we have 6 machines with the following details:

Host Name	IP	Services	Data Path
host1	192.168.1.101	PD1 & TiDB	/data
host2	192.168.1.102	PD2	/data
host3	192.168.1.103	PD3	/data
host4	192.168.1.104	TiKV1	/data
host5	192.168.1.105	TiKV2	/data
host6	192.168.1.106	TiKV3	/data

1. Start PD

Start PD1 on the **host1**

```
docker run -d --name pd1 \
  -p 2379:2379 \
  -p 2380:2380 \
  -v /etc/localtime:/etc/localtime:ro \
  -v /data:/data \
  pingcap/pd:latest \
  --name="pd1" \
  --data-dir="/data/pd1" \
  --client-urls="http://0.0.0.0:2379" \
  --advertise-client-urls="http://192.168.1.101:2379" \
  --peer-urls="http://0.0.0.0:2380" \
  --advertise-peer-urls="http://192.168.1.101:2380" \
  --initial-cluster="pd1=http://192.168.1.101:2380,pd2=http
    ↪ ://192.168.1.102:2380,pd3=http://192.168.1.103:2380"
```

Start PD2 on the **host2**

```
docker run -d --name pd2 \
  -p 2379:2379 \
  -p 2380:2380 \
  -v /etc/localtime:/etc/localtime:ro \
  -v /data:/data \
  pingcap/pd:latest \
  --name="pd2" \
  --data-dir="/data/pd2" \
  --client-urls="http://0.0.0.0:2379" \
  --advertise-client-urls="http://192.168.1.102:2379" \
  --peer-urls="http://0.0.0.0:2380" \
  --advertise-peer-urls="http://192.168.1.102:2380" \
  --initial-cluster="pd1=http://192.168.1.101:2380,pd2=http
    ↪ ://192.168.1.102:2380,pd3=http://192.168.1.103:2380"
```

Start PD3 on the **host3**

```
docker run -d --name pd3 \
  -p 2379:2379 \
  -p 2380:2380 \
  -v /etc/localtime:/etc/localtime:ro \
  -v /data:/data \
  pingcap/pd:latest \
  --name="pd3" \
  --data-dir="/data/pd3" \
  --client-urls="http://0.0.0.0:2379" \
```

```
  --advertise-client-urls="http://192.168.1.103:2379" \
  --peer-urls="http://0.0.0.0:2380" \
  --advertise-peer-urls="http://192.168.1.103:2380" \
  --initial-cluster="pd1=http://192.168.1.101:2380,pd2=http
     ↪ ://192.168.1.102:2380,pd3=http://192.168.1.103:2380"
```

2. Start TiKV

Start TiKV1 on the **host4**

```
docker run -d --name tikv1 \
  -p 20160:20160 \
  -v /etc/localtime:/etc/localtime:ro \
  -v /data:/data \
  pingcap/tikv:latest \
  --addr="0.0.0.0:20160" \
  --advertise-addr="192.168.1.104:20160" \
  --data-dir="/data/tikv1" \
  --pd="192.168.1.101:2379,192.168.1.102:2379,192.168.1.103:2379"
```

Start TiKV2 on the **host5**

```
docker run -d --name tikv2 \
  -p 20160:20160 \
  -v /etc/localtime:/etc/localtime:ro \
  -v /data:/data \
  pingcap/tikv:latest \
  --addr="0.0.0.0:20160" \
  --advertise-addr="192.168.1.105:20160" \
  --data-dir="/data/tikv2" \
  --pd="192.168.1.101:2379,192.168.1.102:2379,192.168.1.103:2379"
```

Start TiKV3 on the **host6**

```
docker run -d --name tikv3 \
  -p 20160:20160 \
  -v /etc/localtime:/etc/localtime:ro \
  -v /data:/data \
  pingcap/tikv:latest \
  --addr="0.0.0.0:20160" \
  --advertise-addr="192.168.1.106:20160" \
  --data-dir="/data/tikv3" \
  --pd="192.168.1.101:2379,192.168.1.102:2379,192.168.1.103:2379"
```

3. Start TiDB

Start TiDB on the **host1**

```
docker run -d --name tidb \
  -p 4000:4000 \
  -p 10080:10080 \
  -v /etc/localtime:/etc/localtime:ro \
  pingcap/tidb:latest \
  --store=tikv \
  --path="192.168.1.101:2379,192.168.1.102:2379,192.168.1.103:2379"
```

4. Use the MySQL client to connect to TiDB

Install the MySQL client on **host1** and run:

```
mysql -h 127.0.0.1 -P 4000 -u root -D test
```

```
show databases;
```

```
+--------------------+
| Database           |
+--------------------+
| INFORMATION_SCHEMA |
| PERFORMANCE_SCHEMA |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

How to customize the configuration file

The TiKV and PD can be started with a specified configuration file, which includes some advanced parameters, for the performance tuning.

Assume that the path to configuration file of PD and TiKV on the host is `/path/to/config/pd.toml` and `/path/to/config/tikv.toml`

You can start TiKV and PD as follows:

```
docker run -d --name tikv1 \
  -p 20160:20160 \
  -v /etc/localtime:/etc/localtime:ro \
  -v /data:/data \
  -v /path/to/config/tikv.toml:/tikv.toml:ro \
  pingcap/tikv:latest \
  --addr="0.0.0.0:20160" \
  --advertise-addr="192.168.1.104:20160" \
  --data-dir="/data/tikv1" \
  --pd="192.168.1.101:2379,192.168.1.102:2379,192.168.1.103:2379" \
  --config="/tikv.toml"
```

```
docker run -d --name pd1 \
  -p 2379:2379 \
  -p 2380:2380 \
  -v /etc/localtime:/etc/localtime:ro \
  -v /data:/data \
  -v /path/to/config/pd.toml:/pd.toml:ro \
  pingcap/pd:latest \
  --name="pd1" \
  --data-dir="/data/pd1" \
  --client-urls="http://0.0.0.0:2379" \
  --advertise-client-urls="http://192.168.1.101:2379" \
  --peer-urls="http://0.0.0.0:2380" \
  --advertise-peer-urls="http://192.168.1.101:2380" \
  --initial-cluster="pd1=http://192.168.1.101:2380,pd2=http
     ↪ ://192.168.1.102:2380,pd3=http://192.168.1.103:2380" \
  --config="/pd.toml"
```

3.2.4 Geographic Redundancy

3.2.4.1 Cross-DC Deployment Solutions

As a NewSQL database, TiDB excels in the best features of the traditional relational database and the scalability of the NoSQL database and is of course, highly available across data centers (hereinafter referred to as DC). This document is to introduce different deployment solutions in cross-DC environment.

3.2.4.1.1 3-DC deployment solution

TiDB, TiKV and PD are distributed among 3 DCs, which is the most common deployment solution with the highest availability.

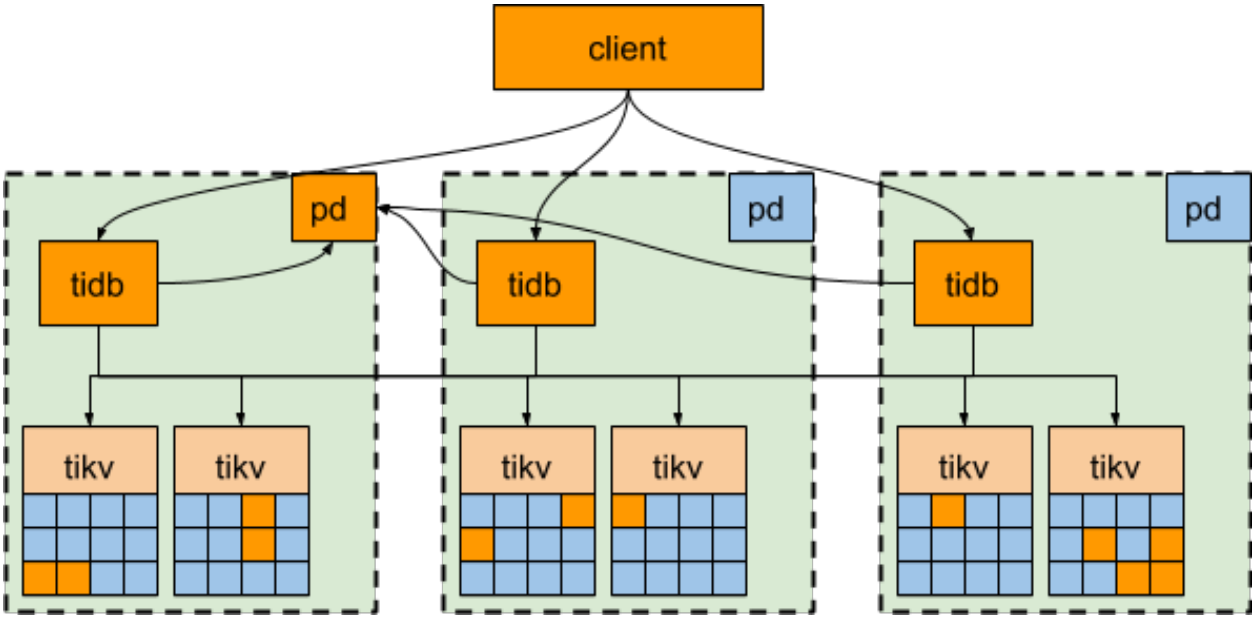

Figure 26: 3-DC Deployment Architecture

Advantages

All the replicas are distributed among 3 DCs. Even if one DC is down, the other 2 DCs will initiate leader election and resume service within a reasonable amount of time (within 20s in most cases) and no data is lost. See the following diagram for more information:

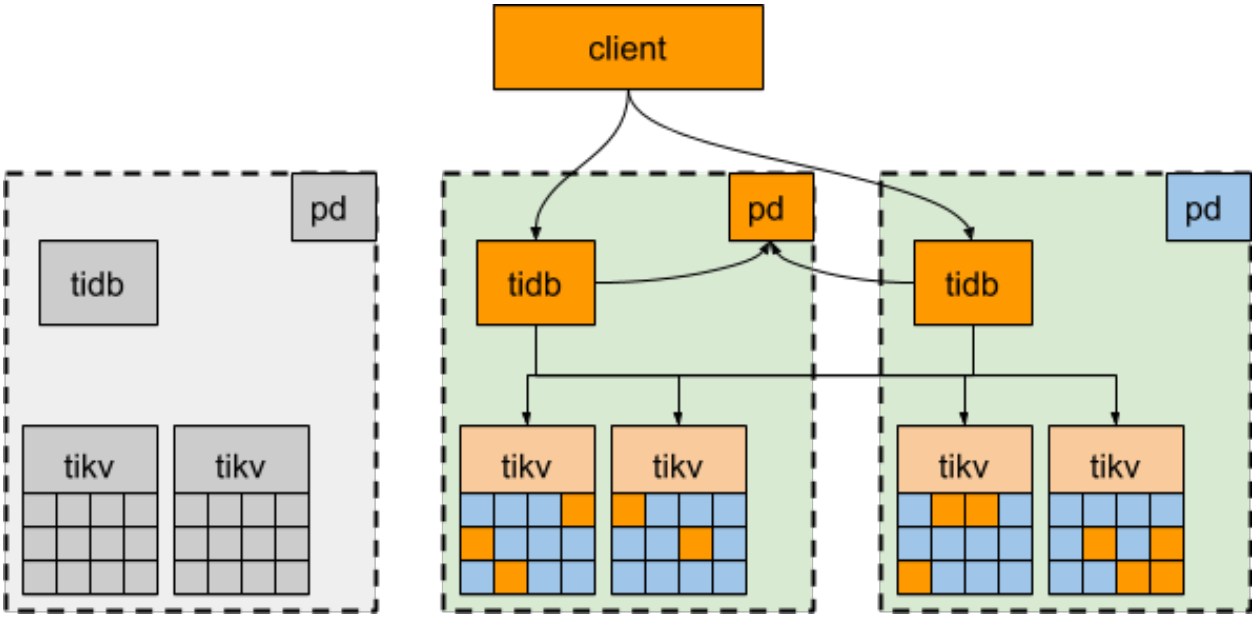

Figure 27: Disaster Recovery for 3-DC Deployment

Disadvantages

The performance is greatly limited by the network latency.

- For writes, all the data has to be replicated to at least 2 DCs. Because TiDB uses 2-phase commit for writes, the write latency is at least twice the latency of the network between two DCs.
- The read performance will also suffer if the leader is not in the same DC as the TiDB node with the read request.
- Each TiDB transaction needs to obtain TimeStamp Oracle (TSO) from the PD leader. So if TiDB and PD leader are not in the same DC, the performance of the transactions will also be impacted by the network latency because each transaction with write request will have to get TSO twice.

Optimizations

If not all of the three DCs need to provide service to the applications, you can dispatch all the requests to one DC and configure the scheduling policy to migrate all the TiKV Region leader and PD leader to the same DC, as what we have done in the following test. In this way, neither obtaining TSO or reading TiKV Regions will be impacted by the network latency between DCs. If this DC is down, the PD leader and Region leader will be automatically elected in other surviving DCs, and you just need to switch the requests to the DC that are still online.

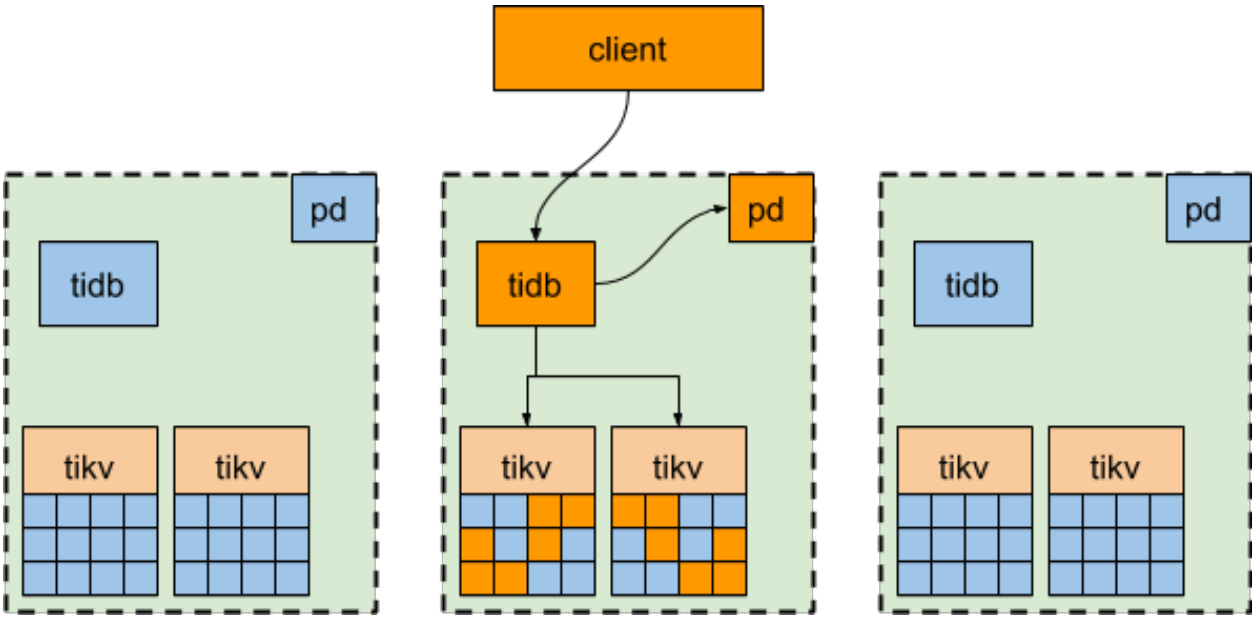

Figure 28: Read Performance Optimized 3-DC Deployment

3.2.4.1.2 3-DC in 2 cities deployment solution

This solution is similar to the previous 3-DC deployment solution and can be considered as an optimization based on the business scenario. The difference is that the distance between the 2 DCs within the same city is short and thus the latency is very low. In this case, we can dispatch the requests to the two DCs within the same city and configure the TiKV leader and PD leader to be in the 2 DCs in the same city.

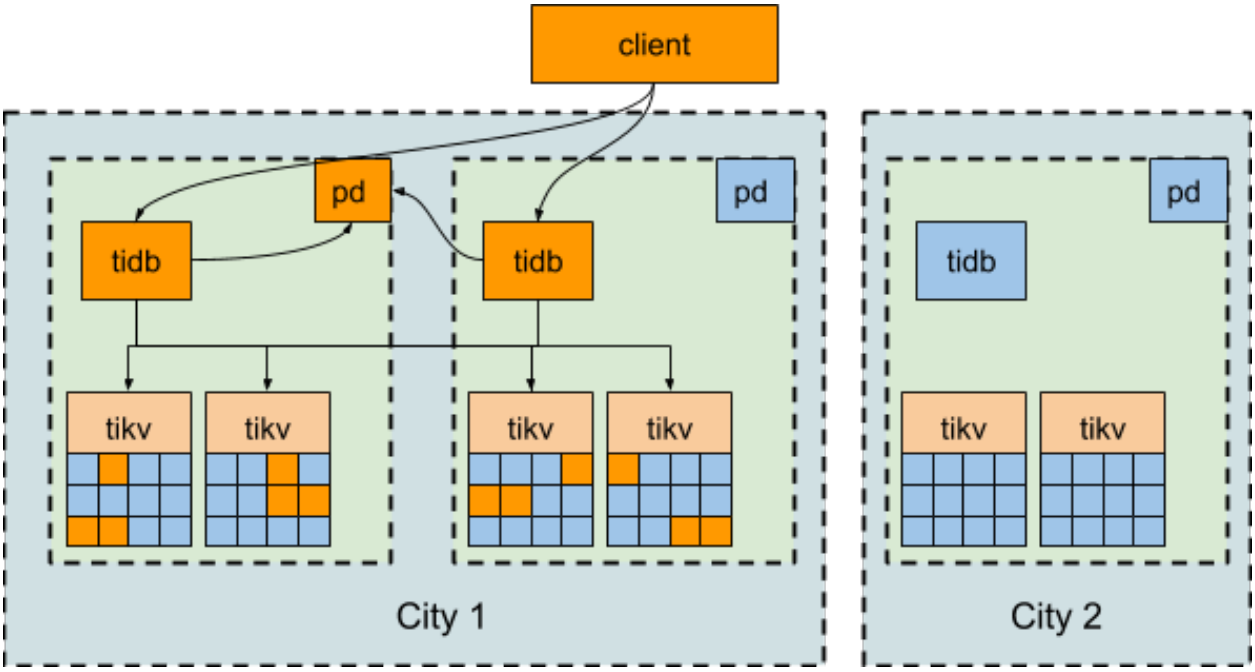

Figure 29: 2-DC in 2 Cities Deployment Architecture

Compared with the 3-DC deployment, the 3-DC in 2 cities deployment has the following advantages:

1. Better write performance.
2. Better usage of the resources because 2 DCs can provide services to the applications.
3. Even if one DC is down, the TiDB cluster will be still available and no data is lost.

However, the disadvantage is that if the 2 DCs within the same city goes down, whose probability is higher than that of the outage of 2 DCs in 2 cities, the TiDB cluster will not be available and some of the data will be lost.

3.2.4.1.3 2-DC + Binlog replication deployment solution

The 2-DC + Binlog replication is similar to the MySQL Source-Replica solution. 2 complete sets of TiDB clusters (each complete set of the TiDB cluster includes TiDB, PD and TiKV) are deployed in 2 DCs, one acts as the primary and one as the secondary. Under

normal circumstances, the primary DC handles all the requests and the data written to the primary DC is asynchronously written to the secondary DC via Binlog.

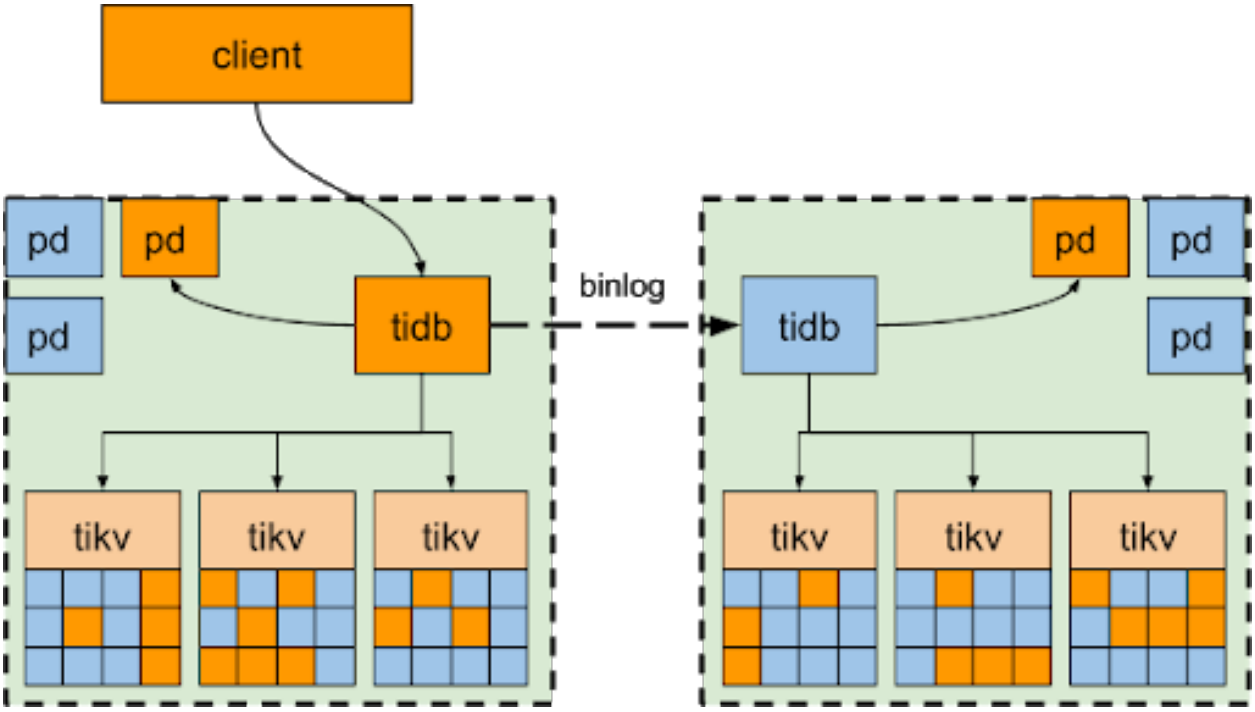

Figure 30: Data Replication in 2-DC in 2 Cities Deployment

If the primary DC goes down, the requests can be switched to the secondary cluster. Similar to MySQL, some data might be lost. But different from MySQL, this solution can ensure the high availability within the same DC: if some nodes within the DC are down, the online workloads won't be impacted and no manual efforts are needed because the cluster will automatically re-elect leaders to provide services.

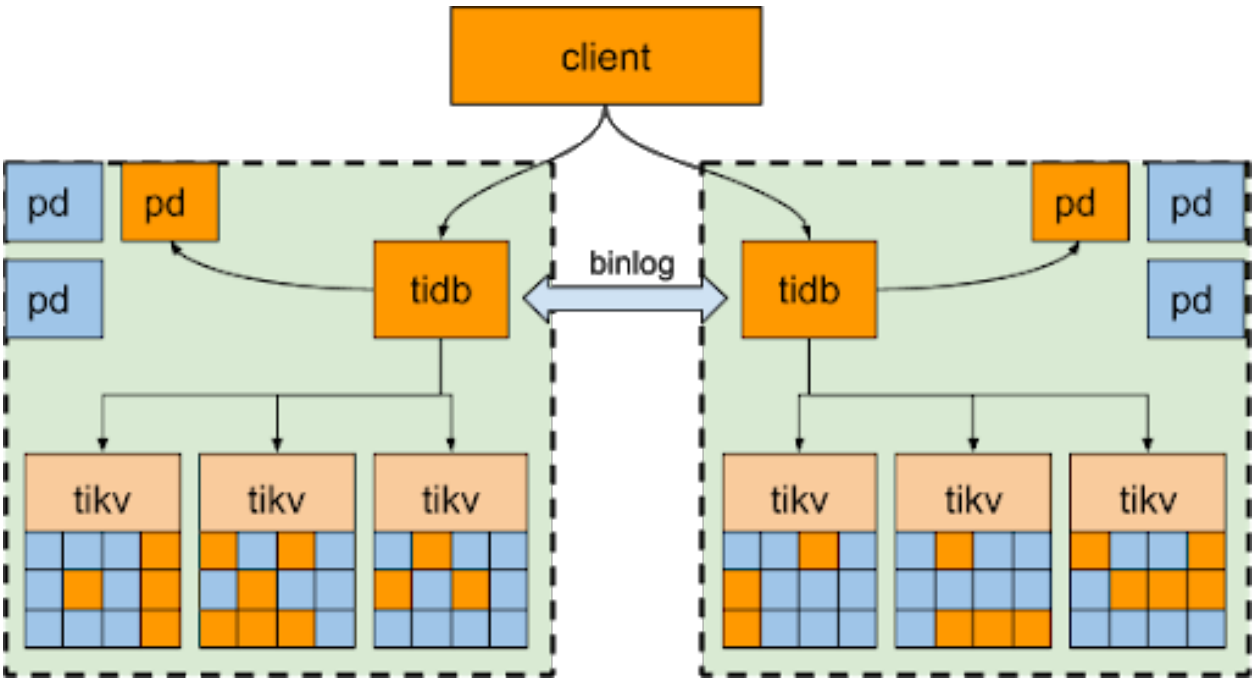

Figure 31: 2-DC as a Mutual Backup Deployment

Some of our production users also adopt the 2-DC multi-active solution, which means:

1. The application requests are separated and dispatched into 2 DCs.
2. Each DC has 1 cluster and each cluster has two databases: A primary database to serve part of the application requests and a secondary database to act as the backup of the other DC's primary database. Data written into the primary database is replicated via Binlog to the secondary database in the other DC, forming a loop of backup.

Please be noted that for the 2-DC + Binlog replication solution, data is asynchronously replicated via Binlog. If the network latency between 2 DCs is too high, the data in the secondary cluster will fall much behind of the primary cluster. If the primary cluster goes down, some data will be lost and it cannot be guaranteed the lost data is within 5 minutes.

3.2.4.1.4 Overall analysis for HA and DR

For the 3-DC deployment solution and 3-DC in 2 cities solution, we can guarantee that the cluster will automatically recover, no human interference is needed and that the data is strongly consistent even if any one of the 3 DCs goes down. All the scheduling policies are to tune the performance, but availability is the top 1 priority instead of performance in case of an outage.

For 2-DC + Binlog replication solution, we can guarantee that the cluster will automatically recover, no human interference is needed and that the data is strongly consistent even

if any some of the nodes within the primary cluster go down. When the entire primary cluster goes down, manual efforts will be needed to switch to the secondary and some data will be lost. The amount of the lost data depends on the network latency and is decided by the network condition.

3.2.4.2 Schedule Replicas by Topology Labels

To improve the high availability and disaster recovery capability of TiDB clusters, it is recommended that TiKV nodes are physically scattered as much as possible. For example, TiKV nodes can be distributed on different racks or even in different data centers. According to the topology information of TiKV, the PD scheduler automatically performs scheduling at the background to isolate each replica of a Region as much as possible, which maximizes the capability of disaster recovery.

To make this mechanism effective, you need to properly configure TiKV and PD so that the topology information of the cluster, especially the TiKV location information, is reported to PD during deployment. Before you begin, see Deploy TiDB Using TiDB Ansible first.

3.2.4.2.1 Configure `labels` based on the cluster topology

Configure `labels` for TiKV

You can use the command-line flag or set the TiKV configuration file to bind some attributes in the form of key-value pairs. These attributes are called `labels`. After TiKV is started, it reports its `labels` to PD so users can identify the location of TiKV nodes.

Assume that the topology has three layers: zone > rack > host, and you can use these labels (zone, rack, host) to set the TiKV location in one of the following methods:

- Use the command-line flag:

```
tikv-server --labels zone=<zone>,rack=<rack>,host=<host>
```

- Configure in the TiKV configuration file:

```
[server]
labels = "zone=<zone>,rack=<rack>,host=<host>"
```

Configure `location-labels` for PD

According to the description above, the label can be any key-value pair used to describe TiKV attributes. But PD cannot identify the location-related labels and the layer relationship of these labels. Therefore, you need to make the following configuration for PD to understand the TiKV node topology.

- If the PD cluster is not initialized, configure `location-labels` in the PD configuration file:

```
[replication]
location-labels = ["zone", "rack", "host"]
```

- If the PD cluster is already initialized, use the pd-ctl tool to make online changes:

```
pd-ctl config set location-labels zone,rack,host
```

The `location-labels` configuration is an array of strings, and each item corresponds to the key of TiKV `labels`. The sequence of each key represents the layer relationship of different labels.

> **Note:**
>
> You must configure `location-labels` for PD and `labels` for TiKV at the same time for the configurations to take effect. Otherwise, PD does not perform scheduling according to the topology.

Configure a cluster using TiDB Ansible

When using TiDB Ansible to deploy a cluster, you can directly configure the TiKV location in the `inventory.ini` file. `tidb-ansible` will generate the corresponding TiKV and PD configuration files during deployment.

In the following example, a two-layer topology of `zone/host` is defined. The TiKV nodes of the cluster are distributed among three zones, each zone with two hosts. In z1, two TiKV instances are deployed per host. In z2 and z3, one TiKV instance is deployed per host.

```
[tikv_servers]
#### z1
tikv-1 labels="zone=z1,host=h1"
tikv-2 labels="zone=z1,host=h1"
tikv-3 labels="zone=z1,host=h2"
tikv-4 labels="zone=z1,host=h2"
#### z2
tikv-5 labels="zone=z2,host=h1"
tikv-6 labels="zone=z2,host=h2"
#### z3
tikv-7 labels="zone=z3,host=h1"
tikv-8 labels="zone=z3,host=h2"

[pd_servers:vars]
location_labels = ["zone", "host"]
```

3.2.4.2.2 PD schedules based on topology label

PD schedules replicas according to the label layer to make sure that different replicas of the same data are scattered as much as possible.

Take the topology in the previous section as an example.

Assume that the number of cluster replicas is 3 (`max-replicas=3`). Because there are 3 zones in total, PD ensures that the 3 replicas of each Region are respectively placed in z1, z2, and z3. In this way, the TiDB cluster is still available when one data center fails.

Then, assume that the number of cluster replicas is 5 (`max-replicas=5`). Because there are only 3 zones in total, PD cannot guarantee the isolation of each replica at the zone level. In this situation, the PD scheduler will ensure replica isolation at the host level. In other words, multiple replicas of a Region might be distributed in the same zone but not on the same host.

In the case of the 5-replica configuration, if z3 fails or is isolated as a whole, and cannot be recovered after a period of time (controlled by `max-store-down-time`), PD will make up the 5 replicas through scheduling. At this time, only 3 hosts are available. This means that host-level isolation cannot be guaranteed and that multiple replicas might be scheduled to the same host.

In summary, PD maximizes the disaster recovery of the cluster according to the current topology. Therefore, if you want to achieve a certain level of disaster recovery, deploy more machines on different sites according to the topology than the number of `max-replicas`.

3.2.5 Data Migration with Ansible

3.3 Configure

3.3.1 Time Zone Support

The time zone in TiDB is decided by the global `time_zone` system variable and the session `time_zone` system variable. The default value of `time_zone` is `SYSTEM`. The actual time zone corresponding to `System` is configured when the TiDB cluster bootstrap is initialized. The detailed logic is as follows:

- Prioritize the use of the `TZ` environment variable.
- If the `TZ` environment variable fails, extract the time zone from the actual soft link address of `/etc/localtime`.
- If both of the above methods fail, use `UTC` as the system time zone.

You can use the following statement to set the global server `time_zone` value at runtime:

```
SET GLOBAL time_zone = timezone;
```

Each client has its own time zone setting, given by the session `time_zone` variable. Initially, the session variable takes its value from the global `time_zone` variable, but the client can change its own time zone with this statement:

```
SET time_zone = timezone;
```

You can use the following statement to view the current values of the global and client-specific time zones:

```
SELECT @@global.time_zone, @@session.time_zone;
```

To set the format of the value of the `time_zone`:

- The value 'SYSTEM' indicates that the time zone should be the same as the system time zone.
- The value can be given as a string indicating an offset from UTC, such as '+10:00' or '-6:00'.
- The value can be given as a named time zone, such as 'Europe/Helsinki', 'US/Eastern', or 'MET'.

The current session time zone setting affects the display and storage of time values that are zone-sensitive. This includes the values displayed by functions such as `NOW()` or `CURTIME()`,

> **Note:**
>
> Only the values of the Timestamp data type is affected by time zone. This is because the Timestamp data type uses the literal value + time zone information. Other data types, such as Datetime/Date/Time, do not have time zone information, thus their values are not affected by the changes of time zone.

```
create table t (ts timestamp, dt datetime);
```

```
Query OK, 0 rows affected (0.02 sec)
```

```
set @@time_zone = 'UTC';
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
insert into t values ('2017-09-30 11:11:11', '2017-09-30 11:11:11');
```

```
Query OK, 1 row affected (0.00 sec)
```

```
set @@time_zone = '+8:00';
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
select * from t;
```

```
+---------------------|---------------------+
| ts                  | dt                  |
+---------------------|---------------------+
| 2017-09-30 19:11:11 | 2017-09-30 11:11:11 |
+---------------------|---------------------+
1 row in set (0.00 sec)
```

In this example, no matter how you adjust the value of the time zone, the value of the Datetime data type is not affected. But the displayed value of the Timestamp data type changes if the time zone information changes. In fact, the value that is stored in the storage does not change, it's just displayed differently according to different time zone setting.

Note:

- Time zone is involved during the conversion of the value of Timestamp and Datetime, which is handled based on the current `time_zone` of the session.
- For data migration, you need to pay special attention to the time zone setting of the primary database and the secondary database.

3.3.2 TiDB Memory Control

Currently, TiDB can track the memory quota of a single SQL query and take actions to prevent OOM (out of memory) or troubleshoot OOM when the memory usage exceeds a specific threshold value. In the TiDB configuration file, you can configure the options as below to control TiDB behaviors when the memory quota exceeds the threshold value:

```
### Valid options: ["log", "cancel"]
oom-action = "log"
```

- If the configuration item above uses "log", when the memory quota of a single SQL query exceeds the threshold value which is controlled by the `tidb_mem_quota_query` variable, TiDB prints an entry of log. Then the SQL query continues to be executed. If OOM occurs, you can find the corresponding SQL query in the log.

- If the configuration item above uses "cancel", when the memory quota of a single SQL query exceeds the threshold value, TiDB stops executing the SQL query immediately and returns an error to the client. The error information clearly shows the memory usage of each physical execution operator that consumes much memory in the SQL execution process.

3.3.2.1 Configure the memory quota of a query

In the configuration file, you can set the default Memory Quota for each Query. The following example sets it to 32GB:

```
mem-quota-query = 34359738368
```

In addition, you can control the memory quota of a query using the following session variables. Generally, you only need to configure `tidb_mem_quota_query`. Other variables are used for advanced configuration which most users do not need to care about.

Variable Name	Description	Unit	Default Value
tidb_mem_quota_query	Control the memory quota of a query	Byte	32 << 30
tidb_mem_quota_hashjoin	Control the memory quota of "HashJoinExec"	Byte	32 << 30
tidb_mem_quota_mergejoin	Control the memory quota of "MergeJoinExec"	Byte	32 << 30
tidb_mem_quota_sort	Control the memory quota of "SortExec"	Byte	32 << 30
tidb_mem_quota_topn	Control the memory quota of "TopNExec"	Byte	32 << 30
tidb_mem_quota_indexlookupreader	Control the memory quota of "IndexLookUpExecutor"	Byte	32 << 30
tidb_mem_quota_indexlookupjoin	Control the memory quota of "IndexLookUpJoin"	Byte	32 << 30
tidb_mem_quota_nestedloopapply	Control the memory quota of "NestedLoopApplyExec"	Byte	32 << 30

Some usage examples:

```
-- Set the threshold value of memory quota for a single SQL query to 8GB:
set @@tidb_mem_quota_query = 8 << 30;

-- Set the threshold value of memory quota for a single SQL query to 8MB:
set @@tidb_mem_quota_query = 8 << 20;

-- Set the threshold value of memory quota for a single SQL query to 8KB:
```

```
set @@tidb_mem_quota_query = 8 << 10;
```

3.3.3 Store Limit

Store Limit is a feature of PD, introduced in TiDB 3.0. It is designed to control the scheduling speed in a finer manner for better performance in different scenarios.

3.3.3.1 Implementation principles

PD performs scheduling at the unit of operator. An operator might contain several scheduling operations. For example:

```
"replace-down-replica {mv peer: store [2] to [3]} (kind:region,replica,
    ↪ region:10(4,5), createAt:2020-05-18 06:40:25.775636418 +0000 UTC m
    ↪ =+2168762.679540369, startAt:2020-05-18 06:40:25.775684648 +0000 UTC
    ↪ m=+2168762.679588599, currentStep:0, steps:[add learner peer 20 on
    ↪ store 3, promote learner peer 20 on store 3 to voter, remove peer on
    ↪ store 2])"
```

In this above example, the `replace-down-replica` operator contains the following specific operations:

1. Add a learner peer with the ID `20` to `store 3`.
2. Promote the learner peer with the ID `20` on `store 3` to a voter.
3. Delete the peer on `store 2`.

Store Limit achieves the store-level speed limit by maintaining a mapping from store IDs to token buckets in memory. The different operations here correspond to different token buckets. Currently, Store Limit only supports limiting the speed of adding learners/peers.

Every time an operator is generated, it checks whether enough tokens exist in the token buckets for its operations. If yes, the operator is added to the scheduling queue, and the corresponding token is taken from the token bucket. Otherwise, the operator is abandoned. Because the token bucket replenishes tokens at a fixed rate, the speed limit is thus achieved.

Store Limit is different from other limit-related parameters in PD (such as `region-` ↪ `schedule-limit` and `leader-schedule-limit`) in that it mainly limits the consuming speed of operators, while other parameters limits the generating speed of operators. Before introducing the Store Limit feature, the speed limit of scheduling is mostly at the global scope. Therefore, even if the global speed is limited, it is still possible that the scheduling operations are concentrated on some stores, affecting the performance of the cluster. By limiting the speed at a finer level, Store Limit can better control the scheduling behavior.

3.3.3.2 Usage

The parameters of Store Limit can be configured using `pd-ctl`.

3.3.3.2.1 View setting of the current store

To view the limit setting of the current store, run the following commands:

```
store limit                    // Shows the speed limit of adding learners/
    ↪ peers in all stores.
```

3.3.3.2.2 Set limit for all stores

To set the speed limit for all stores, run the following commands:

```
stores set limit 5              // All stores can at most add 5 learners/
    ↪ peers per minute.
```

3.3.3.2.3 Set limit for a single store

To set the speed limit for a single store, run the following commands:

```
store limit 1 5                // store 1 can at most add 5 learners/peers
    ↪ per minute.
```

3.3.3.2.4 Persist store limit modification

Because the store limit is a mapping in the memory, the above modification is reset after the leader is switched or PD is restarted. If you want to persist the modification, run the following command:

```
config set store-balance-rate 20 // All stores can at most add 20 learners/
    ↪ peers per minute.
```

3.4 Secure

3.4.1 Transport Layer Security (TLS)

3.4.1.1 Enable TLS for MySQL Clients

It is recommended to use the encrypted connection to ensure data security because non-encrypted connection might lead to information leak.

The TiDB server supports the encrypted connection based on the TLS (Transport Layer Security). The protocol is consistent with MySQL encrypted connections and is directly supported by existing MySQL clients such as MySQL operation tools and MySQL drivers. TLS is sometimes referred to as SSL (Secure Sockets Layer). Because the SSL protocol has known security vulnerabilities, TiDB does not support it. TiDB supports the following versions: TLS 1.0, TLS 1.1, and TLS 1.2.

After using an encrypted connection, the connection has the following security properties:

- Confidentiality: the traffic plaintext cannot be eavesdropped
- Integrity: the traffic plaintext cannot be tampered
- Authentication: (optional) the client and the server can verify the identity of both parties to avoid man-in-the-middle attacks

The encrypted connections in TiDB are disabled by default. To use encrypted connections in the client, you must first configure the TiDB server and enable encrypted connections. In addition, similar to MySQL, the encrypted connections in TiDB consist of single optional connection. For a TiDB server with encrypted connections enabled, you can choose to securely connect to the TiDB server through an encrypted connection, or to use a generally unencrypted connection. Most MySQL clients do not use encrypted connections by default, so generally the client is explicitly required to use an encrypted connection.

In short, to use encrypted connections, both of the following conditions must be met:

1. Enable encrypted connections in the TiDB server.
2. The client specifies to use an encrypted connection.

3.4.1.1.1 Configure TiDB to use encrypted connections

See the following desrciptions about the related parameters to enable encrypted connections:

- `ssl-cert`: specifies the file path of the SSL certificate
- `ssl-key`: specifies the private key that matches the certificate
- `ssl-ca`: (optional) specifies the file path of the trusted CA certificate

To enable encrypted connections in the TiDB server, you must specify both of the `ssl-cert` and `ssl-key` parameters in the configuration file when you start the TiDB server. You can also specify the `ssl-ca` parameter for client authentication (see Enable authentication).

All the files specified by the parameters are in PEM (Privacy Enhanced Mail) format. Currently, TiDB does not support the import of a password-protected private key, so it is required to provide a private key file without a password. If the certificate or private key is invalid, the TiDB server starts as usual, but the client cannot connect to the TiDB server through an encrypted connection.

The certificate or key is signed and generated using OpenSSL, or quickly generated using the `mysql_ssl_rsa_setup` tool in MySQL:

```
mysql_ssl_rsa_setup --datadir=./certs
```

This command generates the following files in the `certs` directory:

```
certs
 ├── ca-key.pem
 ├── ca.pem
 ├── client-cert.pem
 ├── client-key.pem
 ├── private_key.pem
 ├── public_key.pem
 ├── server-cert.pem
 └── server-key.pem
```

The corresponding TiDB configuration file parameters are:

```
[security]
ssl-cert = "certs/server-cert.pem"
ssl-key = "certs/server-key.pem"
```

If the certificate parameters are correct, TiDB outputs `secure connection is enabled` when started, otherwise it outputs `secure connection is NOT ENABLED`.

3.4.1.1.2 Configure the MySQL client to use encrypted connections

The client of MySQL 5.7 or later versions attempts to establish an encrypted connection by default. If the server does not support encrypted connections, it automatically returns to unencrypted connections. The client of MySQL earlier than version 5.7 uses the unencrypted connection by default.

You can change the connection behavior of the client using the following `--ssl-mode` parameters:

- `--ssl-mode=REQUIRED`: The client requires an encrypted connection. The connection cannot be established if the server side does not support encrypted connections.
- In the absence of the `--ssl-mode` parameter: The client attempts to use an encrypted connection, but the encrypted connection cannot be established if the server side does not support encrypted connections. Then the client uses an unencrypted connection.
- `--ssl-mode=DISABLED`: The client uses an unencrypted connection.

For more information, see Client-Side Configuration for Encrypted Connections in MySQL.

3.4.1.1.3 Enable authentication

If the `ssl-ca` parameter is not specified in the TiDB server or MySQL client, the client or the server does not perform authentication by default and cannot prevent man-in-the-middle attack. For example, the client might "securely" connect to a disguised client. You can configure the `ssl-ca` parameter for authentication in the server and client. Generally,

you only need to authenticate the server, but you can also authenticate the client to further enhance the security.

- To authenticate the TiDB server from the MySQL client:

1. Specify the `ssl-cert` and `ssl-key` parameters in the TiDB server.
2. Specify the `--ssl-ca` parameter in the MySQL client.
3. Specify the `--ssl-mode` to `VERIFY_CA` at least in the MySQL client.
4. Make sure that the certificate (`ssl-cert`) configured by the TiDB server is signed by the CA specified by the client `--ssl-ca` parameter, otherwise the authentication fails.

- To authenticate the MySQL client from the TiDB server:

1. Specify the `ssl-cert`, `ssl-key`, and `ssl-ca` parameters in the TiDB server.
2. Specify the `--ssl-cert` and `--ssl-key` parameters in the client.
3. Make sure the server-configured certificate and the client-configured certificate are both signed by the `ssl-ca` specified by the server.

- To perform mutual authentication, meet both of the above requirements.

> **Note:**
>
> Currently, it is optional that TiDB server authenticates the client. If the client does not present its identity certificate in the TLS handshake, the TLS connection can also be successfully established.

3.4.1.1.4 Check whether the current connection uses encryption

Use the `SHOW STATUS LIKE "%Ssl%";` statement to get the details of the current connection, including whether encryption is used, the encryption protocol used by encrypted connections, the TLS version number and so on.

See the following example of the result in an encrypted connection. The results change according to different TLS versions or encryption protocols supported by the client.

```
SHOW STATUS LIKE "%Ssl%";
```

```
......
| Ssl_verify_mode | 5                          |
| Ssl_version     | TLSv1.2                    |
| Ssl_cipher      | ECDHE-RSA-AES128-GCM-SHA256 |
......
```

For the official MySQL client, you can also use the `STATUS` or `\s` statement to view the connection status:

```
\s
```

```
...
SSL: Cipher in use is ECDHE-RSA-AES128-GCM-SHA256
...
```

3.4.1.1.5 Supported TLS versions, key exchange protocols, and encryption algorithms

The TLS versions, key exchange protocols and encryption algorithms supported by TiDB are determined by the official Golang libraries.

Supported TLS versions

- TLS 1.0
- TLS 1.1
- TLS 1.2
- TLS 1.3

Supported key exchange protocols and encryption algorithms

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256

3.4.1.2 Enable TLS Authentication

3.4.1.2.1 Overview

This document describes how to enable TLS authentication in the TiDB cluster. The TLS authentication includes the following two conditions:

- The mutual authentication between TiDB components, including the authentication among TiDB, TiKV and PD, between TiKV Control and TiKV, between PD Control and PD, between TiKV peers, and between PD peers. Once enabled, the mutual authentication applies to all components, and it does not support applying to only part of the components.
- The one-way and mutual authentication between the TiDB server and the MySQL Client.

> **Note:**
>
> The authentication between the MySQL Client and the TiDB server uses one set of certificates, while the authentication among TiDB components uses another set of certificates.

3.4.1.2.2 Enable mutual TLS authentication among TiDB components

Prepare certificates

It is recommended to prepare a separate server certificate for TiDB, TiKV and PD, and make sure that they can authenticate each other. The clients of TiDB, TiKV and PD share one client certificate.

You can use multiple tools to generate self-signed certificates, such as `openssl`, `easy-rsa` and `cfssl`.

See an example of generating self-signed certificates using `cfssl`.

Configure certificates

To enable mutual authentication among TiDB components, configure the certificates of TiDB, TiKV and PD as follows.

TiDB

Configure in the configuration file or command line arguments:

```
[security]
#### Path of file that contains list of trusted SSL CAs for connection with
    ↪ cluster components.
cluster-ssl-ca = "/path/to/ca.pem"
#### Path of file that contains X509 certificate in PEM format for
    ↪ connection with cluster components.
```

```
cluster-ssl-cert = "/path/to/tidb-server.pem"
#### Path of file that contains X509 key in PEM format for connection with
    ↪ cluster components.
cluster-ssl-key = "/path/to/tidb-server-key.pem"
```

TiKV

Configure in the configuration file or command line arguments, and set the corresponding URL to https:

```
[security]
#### set the path for certificates. Empty string means disabling secure
    ↪ connections.
ca-path = "/path/to/ca.pem"
cert-path = "/path/to/tikv-server.pem"
key-path = "/path/to/tikv-server-key.pem"
```

PD

Configure in the configuration file or command line arguments, and set the corresponding URL to https:

```
[security]
#### Path of file that contains list of trusted SSL CAs. If set, following
    ↪ four settings shouldn't be empty
cacert-path = "/path/to/ca.pem"
#### Path of file that contains X509 certificate in PEM format.
cert-path = "/path/to/pd-server.pem"
#### Path of file that contains X509 key in PEM format.
key-path = "/path/to/pd-server-key.pem"
```

Now mutual authentication among TiDB components is enabled.

When you connect the server using the client, it is required to specify the client certificate. For example:

```
./pd-ctl -u https://127.0.0.1:2379 --cacert /path/to/ca.pem --cert /path/to/
    ↪ client.pem --key /path/to/client-key.pem
&&
./tikv-ctl --host="127.0.0.1:20160" --ca-path="/path/to/ca.pem" --cert-path=
    ↪ "/path/to/client.pem" --key-path="/path/to/clinet-key.pem"
```

3.4.1.2.3 Enable TLS authentication between the MySQL client and TiDB server

See Use Encrypted Connections.

3.4.2 Generate Self-Signed Certificates

3.4.2.1 Overview

This document describes how to generate self-signed certificates using `cfssl`.

Assume that the topology of the instance cluster is as follows:

Name	Host IP	Services
node1	172.16.10.1	PD1, TiDB1
node2	172.16.10.2	PD2, TiDB2
node3	172.16.10.3	PD3
node4	172.16.10.4	TiKV1
node5	172.16.10.5	TiKV2
node6	172.16.10.6	TiKV3

3.4.2.2 Download `cfssl`

Assume that the host is x86_64 Linux:

```
mkdir ~/bin &&
curl -s -L -o ~/bin/cfssl https://pkg.cfssl.org/R1.2/cfssl_linux-amd64 &&
curl -s -L -o ~/bin/cfssljson https://pkg.cfssl.org/R1.2/cfssljson_linux-
    ↪ amd64 &&
chmod +x ~/bin/{cfssl,cfssljson} &&
export PATH=$PATH:~/bin
```

3.4.2.3 Initialize the certificate authority

To make it easy for modification later, generate the default configuration of `cfssl`:

```
mkdir ~/cfssl &&
cd ~/cfssl &&
cfssl print-defaults config > ca-config.json &&
cfssl print-defaults csr > ca-csr.json
```

3.4.2.4 Generate certificates

3.4.2.4.1 Certificates description

- tidb-server certificate: used by TiDB to authenticate TiDB for other components and clients
- tikv-server certificate: used by TiKV to authenticate TiKV for other components and clients
- pd-server certificate: used by PD to authenticate PD for other components and clients

- client certificate: used to authenticate the clients from PD, TiKV and TiDB, such as `pd-ctl`, `tikv-ctl` and `pd-recover`

3.4.2.4.2 Configure the CA option

Edit `ca-config.json` according to your need:

```
{
    "signing": {
        "default": {
            "expiry": "43800h"
        },
        "profiles": {
            "server": {
                "expiry": "43800h",
                "usages": [
                    "signing",
                    "key encipherment",
                    "server auth",
                    "client auth"
                ]
            },
            "client": {
                "expiry": "43800h",
                "usages": [
                    "signing",
                    "key encipherment",
                    "client auth"
                ]
            }
        }
    }
}
```

Edit `ca-csr.json` according to your need:

```
{
    "CN": "My own CA",
    "key": {
        "algo": "rsa",
        "size": 2048
    },
    "names": [
        {
            "C": "CN",
            "L": "Beijing",
```

```
            "O": "PingCAP",
            "ST": "Beijing"
        }
    ]
}
```

3.4.2.4.3 Generate the CA certificate

```
cfssl gencert -initca ca-csr.json | cfssljson -bare ca -
```

The command above generates the following files:

```
ca-key.pem
ca.csr
ca.pem
```

3.4.2.4.4 Generate the server certificate

The IP address of all components and `127.0.0.1` are included in `hostname`.

```
echo '{"CN":"tidb-server","hosts":[""],"key":{"algo":"rsa","size":2048}}' |
    ↪  cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -
    ↪ profile=server -hostname="172.16.10.1,172.16.10.2,127.0.0.1" - |
    ↪ cfssljson -bare tidb-server
&&
echo '{"CN":"tikv-server","hosts":[""],"key":{"algo":"rsa","size":2048}}' |
    ↪  cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -
    ↪ profile=server -hostname="
    ↪ 172.16.10.4,172.16.10.5,172.16.10.6,127.0.0.1" - | cfssljson -bare
    ↪ tikv-server
&&
echo '{"CN":"pd-server","hosts":[""],"key":{"algo":"rsa","size":2048}}' |
    ↪ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -
    ↪ profile=server -hostname="
    ↪ 172.16.10.1,172.16.10.2,172.16.10.3,127.0.0.1" - | cfssljson -bare pd
    ↪ -server
```

The command above generates the following files:

```
tidb-server-key.pem  tikv-server-key.pem   pd-server-key.pem
tidb-server.csr      tikv-server.csr       pd-server.csr
tidb-server.pem      tikv-server.pem       pd-server.pem
```

3.4.2.4.5 Generate the client certificate

```
echo '{"CN":"client","hosts":[""],"key":{"algo":"rsa","size":2048}}' |
    ↪ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -
    ↪ profile=client -hostname="" - | cfssljson -bare client
```

The command above generates the following files:

```
client-key.pem
client.csr
client.pem
```

3.5 Monitor

3.5.1 TiDB Monitoring Framework Overview

The TiDB monitoring framework adopts two open source projects: Prometheus and Grafana. TiDB uses Prometheus to store the monitoring and performance metrics and Grafana to visualize these metrics.

3.5.1.1 About Prometheus in TiDB

As a time series database, Prometheus has a multi-dimensional data model and flexible query language. As one of the most popular open source projects, Prometheus has been adopted by many companies and organizations and has a very active community. PingCAP is one of the active developers and adopters of Prometheus for monitoring and alerting in TiDB, TiKV and PD.

Prometheus consists of multiple components. Currently, TiDB uses the following of them:

- The Prometheus Server to scrape and store time series data
- The client libraries to customize necessary metrics in the application
- An Alertmanager for the alerting mechanism

The diagram is as follows:

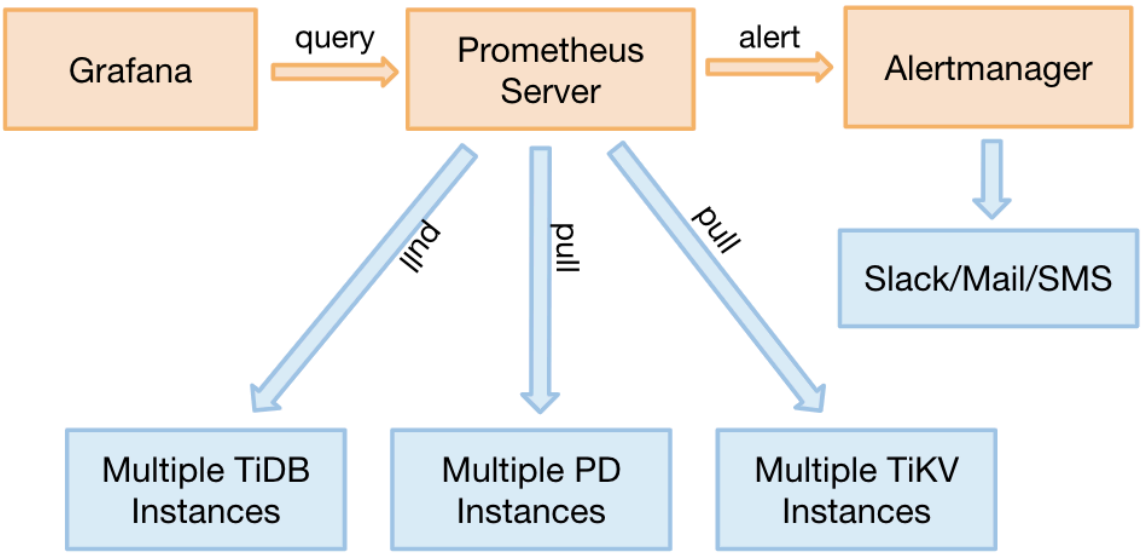

Figure 32: diagram

3.5.1.2 About Grafana in TiDB

Grafana is an open source project for analyzing and visualizing metrics. TiDB uses Grafana to display the performance metrics as follows:

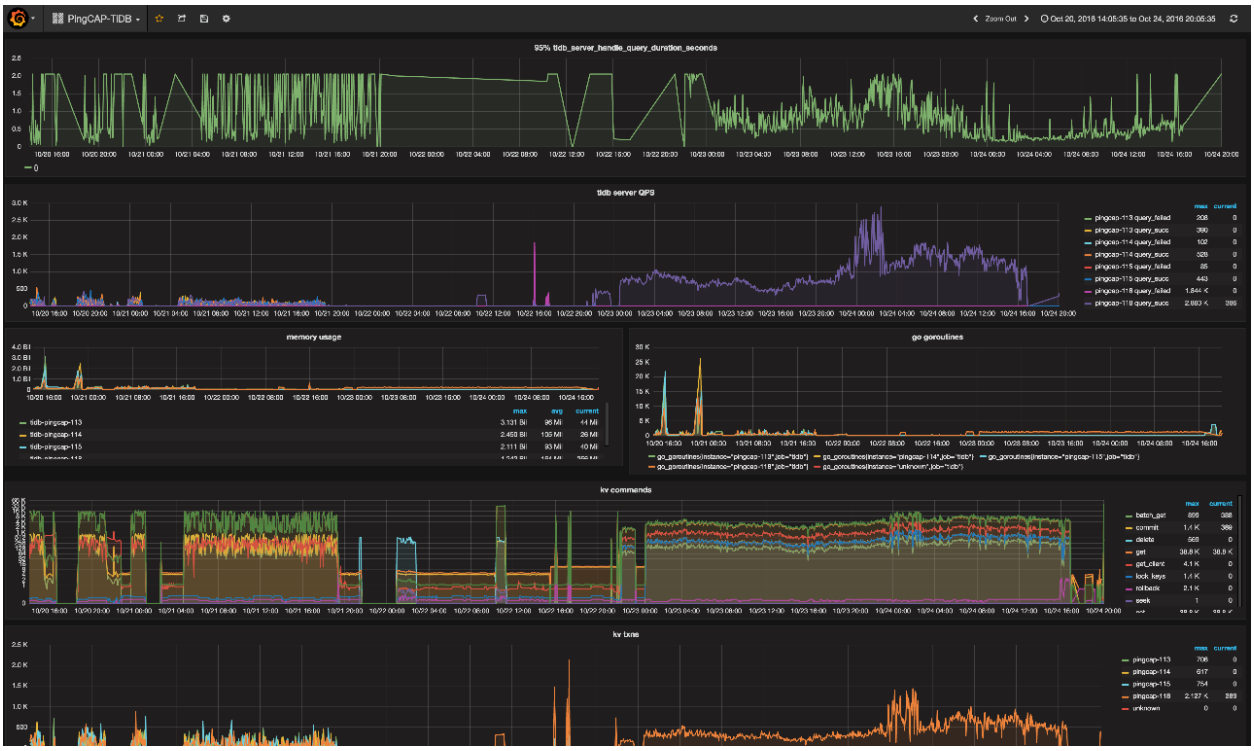

Figure 33: screenshot

3.5.2 Monitor a TiDB Cluster

You can use the following two types of interfaces to monitor the TiDB cluster state:

- The state interface: this interface uses the HTTP interface to get the component information.
- The metrics interface: this interface uses Prometheus to record the detailed information of the various operations in components and views these metrics using Grafana.

3.5.2.1 Use the state interface

The state interface monitors the basic information of a specific component in the TiDB cluster. It can also act as the monitor interface for Keepalive messages. In addition, the state interface for the Placement Driver (PD) can get the details of the entire TiKV cluster.

3.5.2.1.1 TiDB server

- TiDB API address: `http://${host}:${port}`
- Default port: `10080`
- Details about API names: see TiDB HTTP API

The following example uses `http://${host}:${port}/status` to get the current state of the TiDB server and to determine whether the server is alive. The result is returned in JSON format.

```
curl http://127.0.0.1:10080/status
```

```
{
    connections: 0, # The current number of clients connected to the TiDB
        ↪ server.
    version: "5.7.25-TiDB-v3.0.0-beta-250-g778c3f4a5", # The TiDB version
        ↪ number.
    git_hash: "778c3f4a5a716880bcd1d71b257c8165685f0d70" # The Git Hash of
        ↪ the current TiDB code.
}
```

3.5.2.1.2 PD server

- PD API address: `http://${host}:${port}/pd/api/v1/${api_name}`
- Default port: 2379
- Details about API names: see PD API doc

The PD interface provides the state of all the TiKV servers and the information about load balancing. See the following example for the information about a single-node TiKV cluster:

```
curl http://127.0.0.1:2379/pd/api/v1/stores
```

```
{
  "count": 1, # The number of TiKV nodes.
  "stores": [ # The list of TiKV nodes.
    # The details about the single TiKV node.
    {
      "store": {
        "id": 1,
        "address": "127.0.0.1:20160",
        "version": "3.0.0-beta",
        "state_name": "Up"
      },
      "status": {
        "capacity": "20 GiB", # The total capacity.
        "available": "16 GiB", # The available capacity.
        "leader_count": 17,
        "leader_weight": 1,
        "leader_score": 17,
```

```
      "leader_size": 17,
      "region_count": 17,
      "region_weight": 1,
      "region_score": 17,
      "region_size": 17,
      "start_ts": "2019-03-21T14:09:32+08:00", # The starting timestamp.
      "last_heartbeat_ts": "2019-03-21T14:14:22.961171958+08:00", # The
         ↪ timestamp of the last heartbeat.
      "uptime": "4m50.961171958s"
    }
  }
]
```

3.5.2.2 Use the metrics interface

The metrics interface monitors the state and performance of the entire TiDB cluster.

- If you use TiDB Ansible to deploy the TiDB cluster, the monitoring system (Prometheus and Grafana) is deployed at the same time.
- If you use other deployment ways, deploy Prometheus and Grafana before using this interface.

After Prometheus and Grafana are successfully deployed, configure Grafana.

3.5.2.2.1 Deploy Prometheus and Grafana

Assume that the TiDB cluster topology is as follows:

Name	Host IP	Services
Node1	192.168.199.113	PD1, TiDB, node_export, Prometheus, Grafana
Node2	192.168.199.114	PD2, node_export
Node3	192.168.199.115	PD3, node_export
Node4	192.168.199.116	TiKV1, node_export
Node5	192.168.199.117	TiKV2, node_export
Node6	192.168.199.118	TiKV3, node_export

Step 1: Download the binary package

1. Download the package.

```
wget https://download.pingcap.org/prometheus-2.8.1.linux-amd64.tar.gz
wget https://download.pingcap.org/node_exporter-0.17.0.linux-amd64.tar.
   ↪ gz
```

```
wget https://download.pingcap.org/grafana-6.1.6.linux-amd64.tar.gz
```

2. Extract the package.

```
tar -xzf prometheus-2.8.1.linux-amd64.tar.gz
tar -xzf node_exporter-0.17.0.linux-amd64.tar.gz
tar -xzf grafana-6.1.6.linux-amd64.tar.gz
```

Step 2: Start `node_exporter` on Node1, Node2, Node3, and Node4

1. Enter the corresponding directory:

```
cd node_exporter-0.17.0.linux-amd64
```

2. Start the node_exporter service.

```
./node_exporter --web.listen-address=":9100" \
    --log.level="info" &
```

Step 3: Start Prometheus on Node1

1. Edit the Prometheus configuration file:

```
cd prometheus-2.8.1.linux-amd64 &&
vi prometheus.yml
```

```
...

global:
  scrape_interval:   15s  # By default, scrape targets every 15 seconds.
  evaluation_interval: 15s # By default, scrape targets every 15
     ↪ seconds.
  # scrape_timeout is set to the global default value (10s).
  external_labels:
    cluster: 'test-cluster'
    monitor: "prometheus"

scrape_configs:
  - job_name: 'overwritten-nodes'
    honor_labels: true # Do not overwrite job & instance labels.
    static_configs:
    - targets:
      - '192.168.199.113:9100'
      - '192.168.199.114:9100'
```

```
    - '192.168.199.115:9100'
    - '192.168.199.116:9100'
    - '192.168.199.117:9100'
    - '192.168.199.118:9100'

  - job_name: 'tidb'
    honor_labels: true # Do not overwrite job & instance labels.
    static_configs:
    - targets:
      - '192.168.199.113:10080'

  - job_name: 'pd'
    honor_labels: true # Do not overwrite job & instance labels.
    static_configs:
    - targets:
      - '192.168.199.113:2379'
      - '192.168.199.114:2379'
      - '192.168.199.115:2379'

  - job_name: 'tikv'
    honor_labels: true # Do not overwrite job & instance labels.
    static_configs:
    - targets:
      - '192.168.199.116:20180'
      - '192.168.199.117:20180'
      - '192.168.199.118:20180'

...
```

2. Start the Prometheus service:

```
./prometheus \
    --config.file="./prometheus.yml" \
    --web.listen-address=":9090" \
    --web.external-url="http://192.168.199.113:9090/" \
    --web.enable-admin-api \
    --log.level="info" \
    --storage.tsdb.path="./data.metrics" \
    --storage.tsdb.retention="15d" &
```

Step 4: Start Grafana on Node1

1. Edit the Grafana configuration file:

```
cd grafana-6.1.6 &&
vi conf/grafana.ini
```

```
...

[paths]
data = ./data
logs = ./data/log
plugins = ./data/plugins
[server]
http_port = 3000
domain = 192.168.199.113
[database]
[session]
[analytics]
check_for_updates = true
[security]
admin_user = admin
admin_password = admin
[snapshots]
[users]
[auth.anonymous]
[auth.basic]
[auth.ldap]
[smtp]
[emails]
[log]
mode = file
[log.console]
[log.file]
level = info
format = text
[log.syslog]
[event_publisher]
[dashboards.json]
enabled = false
path = ./data/dashboards
[metrics]
[grafana_net]
url = https://grafana.net

...
```

2. Start the Grafana service:

```
./bin/grafana-server \
    --config="./conf/grafana.ini" &
```

3.5.2.2.2 Configure Grafana

This section describes how to configure Grafana.

Step 1: Add a Prometheus data source

1. Log in to the Grafana Web interface.

    - Default address: http://localhost:3000
    - Default account: admin
    - Default password: admin

> **Note:**
> For the **Change Password** step, you can choose **Skip**.

2. In the Grafana sidebar menu, click **Data Source** within the **Configuration**.
3. Click **Add data source**.
4. Specify the data source information.

    - Specify a **Name** for the data source.
    - For **Type**, select **Prometheus**.
    - For **URL**, specify the Prometheus address.
    - Specify other fields as needed.

5. Click **Add** to save the new data source.

Step 2: Import a Grafana dashboard

To import a Grafana dashboard for the PD server, the TiKV server, and the TiDB server, take the following steps respectively:

1. Click the Grafana logo to open the sidebar menu.
2. In the sidebar menu, click **Dashboards** -> **Import** to open the **Import Dashboard** window.
3. Click **Upload .json File** to upload a JSON file (Download TiDB Grafana configuration file).

> **Note:**
> For the TiKV, PD, and TiDB dashboards, the corresponding JSON files are `tikv_summary.json`, `tikv_details.json`, `tikv_trouble_shooting.json`, `pd.json`, `tidb.json`, and `tidb_summary.json`.

4. Click **Load**.
5. Select a Prometheus data source.
6. Click **Import**. A Prometheus dashboard is imported.

3.5.2.2.3 View component metrics

Click **New dashboard** in the top menu and choose the dashboard you want to view.

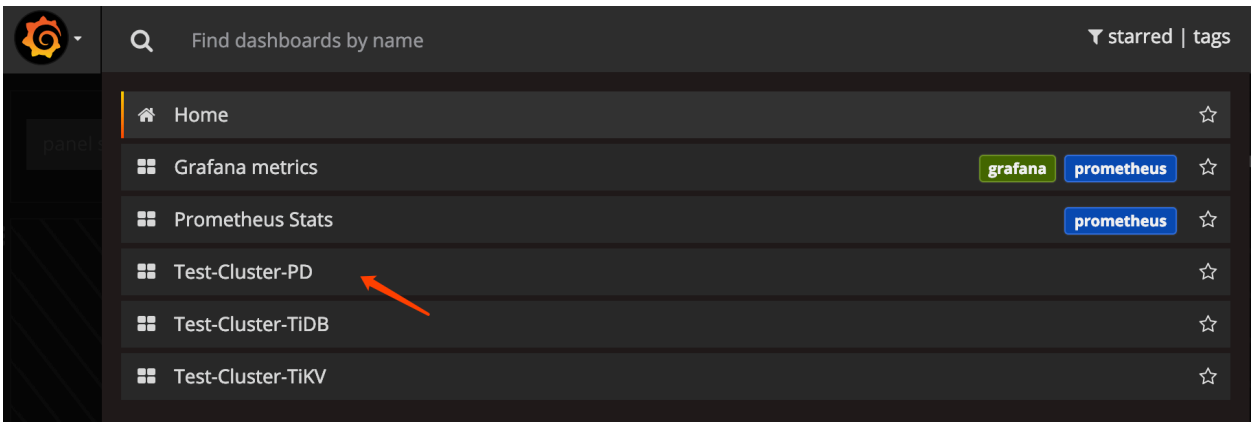

Figure 34: view dashboard

You can get the following metrics for cluster components:

- **TiDB server:**
  - Query processing time to monitor the latency and throughput
  - The DDL process monitoring
  - TiKV client related monitoring
  - PD client related monitoring
- **PD server:**
  - The total number of times that the command executes
  - The total number of times that a certain command fails

  - The duration that a command succeeds
  - The duration that a command fails
  - The duration that a command finishes and returns result

- **TiKV server:**

  - Garbage Collection (GC) monitoring
  - The total number of times that the TiKV command executes
  - The duration that Scheduler executes commands
  - The total number of times of the Raft propose command
  - The duration that Raft executes commands
  - The total number of times that Raft commands fail
  - The total number of times that Raft processes the ready state

3.5.3 Export Grafana Snapshots

Metrics data is important in troubleshooting. When you request remote assistance, sometimes the support staff need to view the Grafana dashboards to diagnose problems. MetricsTool can help export snapshots of Grafana dashboards as local files and visualize these snapshots. You can share these snapshots with outsiders and allow them to accurately read out the graphs, without giving out access to other sensitive information on the Grafana server.

3.5.3.1 Usage

MetricsTool can be accessed from https://metricstool.pingcap.com/. It consists of three sets of tools:

- **Export**: A user script running on the browser's Developer Tool, allowing you to download a snapshot of all visible panels in the current dashboard on any Grafana v6.x.x server.

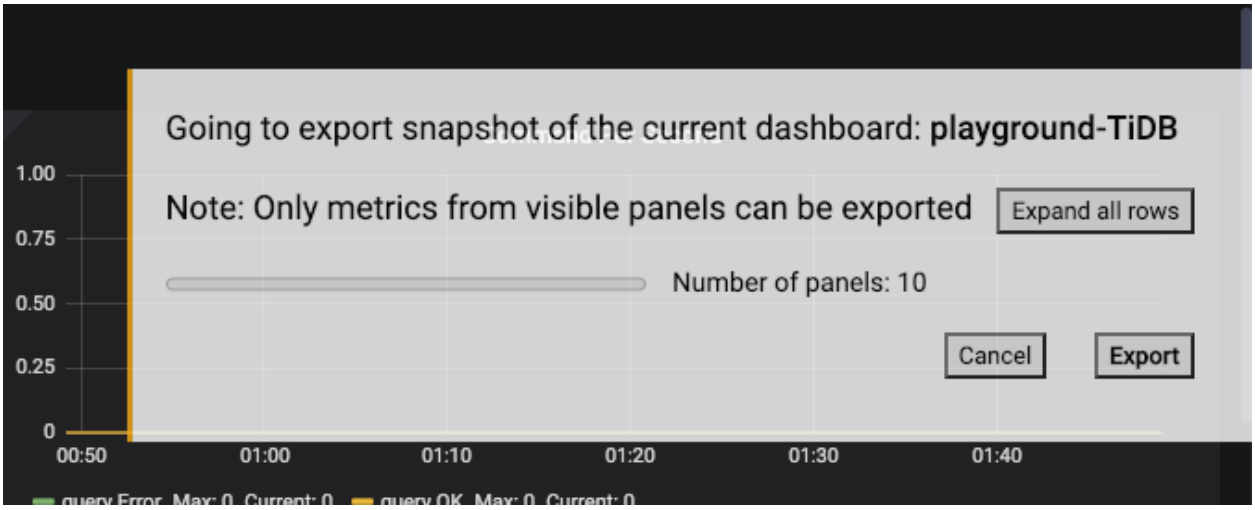

Figure 35: Screenshot of MetricsTool Exporter after running the user script

- **Visualize**: A web page visualizing the exported snapshot files. The visualized snapshots can be operated in the same way as live Grafana dashboards.

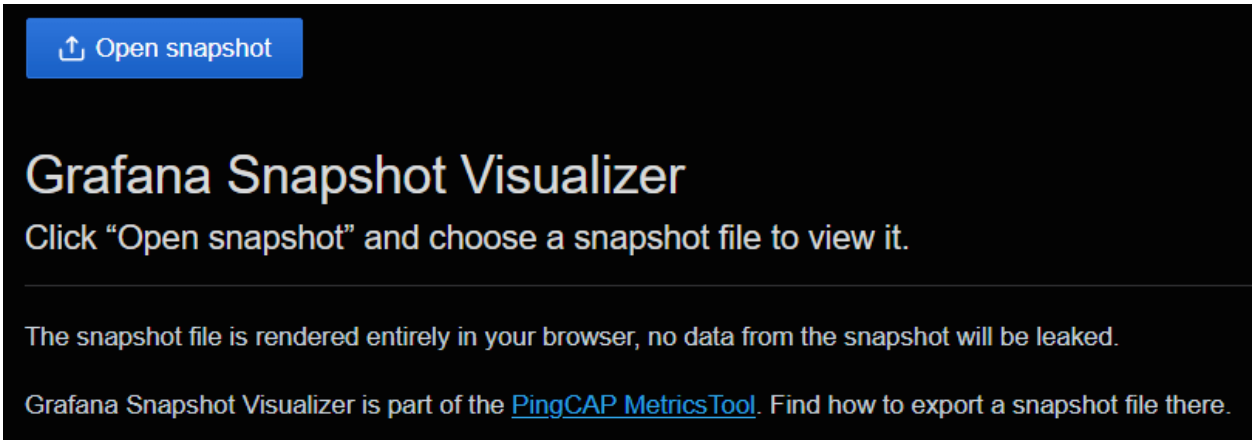

Figure 36: Screenshot of MetricsTool Visualizer

- **Import**: Instructions to import the exported snapshot back into an actual Grafana instance.

3.5.3.2 FAQs

3.5.3.2.1 What is the advantage of this tool compared with screenshot or PDF printing?

The snapshot files exported by MetricsTool contain the actual values when they are taken. And the Visualizer allows you to interact with the rendered graphs as if it is a live Grafana dashboard, supporting operations like toggling series, zooming into a smaller time range, and checking the precise value at a given time. This makes MetricsTool much more powerful than images or PDFs.

3.5.3.2.2 What are included in the snapshot file?

The snapshot file contains the values of all graphs and panels in the selected time range. It does not save the original metrics from the data sources (and thus you cannot edit the query expression in the Visualizer).

3.5.3.2.3 Will the Visualizer save the uploaded snapshot files in PingCAP's servers?

No, the Visualizer parses the snapshot files entirely inside your browser. Nothing will be sent to PingCAP. You are free to view snapshot files received from sensitive sources, and no need to worry about these leaking to third parties through the Visualizer.

3.5.3.2.4 Can it export metrics besides Grafana?

No, we only support Grafana v6.x.x at the moment.

3.5.3.2.5 Will there be problems to execute the script before all metrics are loaded?

No, the script UI will notify you to wait for all metrics to be loaded. However, you can manually skip waiting and export the snapshot in case of some metrics loading for too long.

3.5.3.2.6 Can we share a link to a visualized snapshot?

No, but you can share the snapshot file, with instruction on how to use the Visualizer to view it. If you truly need a world-readable URL, you may also try the public `snapshot.raintank.io` service built into Grafana, but make sure all privacy concerns are cleared before doing so.

3.6 Migrate

3.6.1 TiDB Tools Overview

This document introduces the functionalities of TiDB tools and their relationship.

3.6.1.1 Deploy and operate TiDB in Kubernetes

TiDB Operator is an automatic operation system for TiDB clusters in Kubernetes. It provides full life-cycle management for TiDB including deployment, upgrades, scaling, backup, fail-over, and configuration changes. With TiDB Operator, TiDB can run seamlessly in the Kubernetes clusters deployed on a public or private cloud.

The following are the basics of TiDB Operator:

- TiDB Operator Architecture
- Get Started with TiDB Operator in Kubernetes
- Applicable TiDB versions: v2.1 and above

3.6.1.2 Full data export

Dumpling is a tool for the logical full data export from MySQL or TiDB.

The following are the basics of Dumpling:

- Input: MySQL/TiDB cluster
- Output: SQL/CSV file
- Supported TiDB versions: all versions
- Kubernetes support: No

3.6.1.3 Full data import

TiDB Lightning (Lightning) is a tool used for the full import of large amounts of data into a TiDB cluster. Currently, TiDB Lightning supports reading SQL dump exported via Dumpling or CSV data source.

TiDB Lightning supports two modes:

- `importer`: This mode uses tikv-importer as the backend, which is usually for importing a large amount of data (at the TB level). During the import, the cluster cannot provide services.
- `tidb`: This mode uses TiDB/MySQL as the backend, which is slower than the `importer` mode but can be performed online. It also supports importing data to MySQL.

The following are the basics of TiDB Lightning:

- Input data source:
  - The output file of Dumpling
  - Other compatible CSV file
- Supported TiDB versions: v2.1 or later

- Kubernetes support: Yes. See Quickly restore data into a TiDB cluster in Kubernetes using TiDB Lightning for details.

> **Note:**
>
> The Loader tool is no longer maintained. For scenarios related to Loader, it is recommended that you use the `tidb` mode of TiDB Lighting instead.

3.6.1.4 Backup and restore

Dumpling can be used to back up the TiDB cluster to SQL/CSV files. See Full data export.

TiDB Lightning can be used to restore the SQL/CSV files exported by Dumpling to the TiDB cluster. See Full data import.

3.6.1.5 Incremental data replication

TiDB Binlog is a tool that collects binlog for TiDB clusters and provides near real-time sync and backup. It can be used for incremental data replication between TiDB clusters, such as making a TiDB cluster the secondary cluster of the primary TiDB cluster.

The following are the basics of TiDB Binlog:

- Input: TiDB cluster
- Output: TiDB cluster, MySQL, Kafka or incremental backup files
- Supported TiDB versions: v2.1 or later
- Kubernetes support: Yes. See TiDB Binlog Cluster Operations and TiDB Binlog Drainer Configurations in Kubernetes for details.

3.6.1.6 Data migration

TiDB Data Migration (DM) is an integrated data replication task management platform that supports the full data migration and the incremental data migration from MySQL/-MariaDB to TiDB.

The following are the basics of DM:

- Input: MySQL/MariaDB
- Output: TiDB cluster
- Supported TiDB versions: all versions
- Kubernetes support: No, under development

If the data volume is below the TB level, it is recommended to migrate data from MySQL/MariaDB to TiDB directly using DM. The migration process includes the full data import and export and the incremental data replication.

If the data volume is at the TB level, take the following steps:

1. Use Dumpling to export the full data from MySQL/MariaDB.
2. Use TiDB Lightning to import the data exported in Step 1 to the TiDB cluster.
3. Use DM to migrate the incremental data from MySQL/MariaDB to TiDB.

> **Note:**
>
> The Syncer tool is no longer maintained. For scenarios related to Syncer, it is recommended that you use DM's incremental task mode instead.

3.6.2 Migrate from MySQL

3.6.2.1 Migrate from MySQL SQL Files Using TiDB Lightning

This document describes how to migrate data from MySQL SQL files to TiDB using TiDB Lightning. For details on how to generate MySQL SQL files, refer to Mydumper or Dumpling.

The data migration process described in this document uses TiDB Lightning. The steps are as follows.

3.6.2.1.1 Step 1: Deploy TiDB Lightning

Before you start the migration, deploy TiDB Lightning.

> **Note:**
>
> - If you choose the Importer-backend to import data, you need to deploy `tikv-importer` along with TiDB Lightning. During the import process, the TiDB cluster cannot provide services. This mode imports data quickly, which is suitable for importing a large amount of data (above the TB level).
> - If you choose the TiDB-backend, deploy TiDB Lightning only. The cluster can provide services normally during the import.
> - For detailed differences between the two backend mode, see TiDB Lightning Backend.

3.6.2.1.2 Step 2: Configure data source of TiDB Lightning

This document takes the TiDB-backend as an example. Create the `tidb-lightning.toml` configuration file and add the following major configurations in the file:

1. Set the `data-source-dir` under `[mydumper]` to the path of the MySQL SQL file.

```
[mydumper]
# Data source directory
data-source-dir = "/data/export"
```

> **Note:**
>
> If a corresponding schema already exists in the downstream, set `no-schema=true` to skip the creation of the schema.

2. Add the configuration of the target TiDB cluster.

```
[tidb]
# The target cluster information. Fill in one address of tidb-server.
host = "172.16.31.1"
port = 4000
user = "root"
password = ""
```

For other configurations, see TiDB Lightning Configuration.

3.6.2.1.3 Step 3: Run TiDB Lightning to import data

Run TiDB Lightning to start the import operation. If you start TiDB Lightning by using `nohup` directly in the command line, the program might exit because of the `SIGHUP` signal. Therefore, it is recommended to write `nohup` in a script. For example:

```
#### !/bin/bash
nohup ./tidb-lightning -config tidb-lightning.toml > nohup.out &
```

When the import operation is started, view the progress by the following two ways:

- `grep` the keyword `progress` in logs, which is updated every 5 minutes by default.
- Access the monitoring dashboard. See Monitor TiDB Lightning for details.

3.6.2.2 Migrate from Amazon Aurora MySQL Using DM

To migrate data from MySQL (Amazon Aurora) to TiDB, refer to Migrate from MySQL (Amazon Aurora).

3.6.3 TiDB Lightning CSV Support

TiDB Lightning supports reading CSV (comma-separated values) data source, as well as other delimited format such as TSV (tab-separated values).

3.6.3.1 File name

A CSV file representing a whole table must be named as `db_name.table_name.csv`. This will be restored as a table `table_name` inside the database `db_name`.

If a table spans multiple CSV files, they should be named like `db_name.table_name.003.csv`.

The file extension must be `*.csv`, even if the content is not separated by commas.

3.6.3.2 Schema

CSV files are schema-less. To import them into TiDB, a table schema must be provided. This could be done either by:

- Providing a file named `db_name.table_name-schema.sql` containing the `CREATE TABLE` DDL statement
- Creating the empty tables directly in TiDB in the first place, and then setting `[mydumper] no-schema = true` in `tidb-lightning.toml`.

3.6.3.3 Configuration

The CSV format can be configured in `tidb-lightning.toml` under the `[mydumper.csv]` section. Most settings have a corresponding option in the MySQL `LOAD DATA` statement.

```
[mydumper.csv]
### Separator between fields, should be an ASCII character.
separator = ','
### Quoting delimiter, can either be an ASCII character or empty string.
delimiter = '"'
### Whether the CSV files contain a header.
### If `header` is true, the first line will be skipped.
header = true
### Whether the CSV contains any NULL value.
### If `not-null` is true, all columns from CSV cannot be NULL.
not-null = false
### When `not-null` is false (that is, CSV can contain NULL),
### fields equal to this value will be treated as NULL.
null = '\N'
### Whether to interpret backslash escapes inside fields.
backslash-escape = true
### If a line ends with a separator, remove it.
trim-last-separator = false
```

3.6.3.3.1 `separator`

- Defines the field separator.
- Must be a single ASCII character.
- Common values:
  - `','` for CSV
  - `"\t"` for TSV
- Corresponds to the `FIELDS TERMINATED BY` option in the LOAD DATA statement.

3.6.3.3.2 `delimiter`

- Defines the delimiter used for quoting.
- If `delimiter` is empty, all fields are unquoted.
- Common values:
  - `'"'` quote fields with double-quote, same as RFC 4180
  - `''` disable quoting
- Corresponds to the `FIELDS ENCLOSED BY` option in the `LOAD DATA` statement.

3.6.3.3.3 `header`

- Whether *all* CSV files contain a header row.
- If `header` is true, the first row will be used as the *column names*. If `header` is false, the first row is not special and treated as an ordinary data row.

3.6.3.3.4 `not-null` and `null`

- The `not-null` setting controls whether all fields are non-nullable.
- If `not-null` is false, the string specified by `null` will be transformed to the SQL NULL instead of a concrete value.
- Quoting will not affect whether a field is null.

  For example, with the CSV file:

```
A,B,C
\N,"\N",
```

  In the default settings (`not-null = false; null = '\N'`), the columns `A` and `B` are both converted to NULL after importing to TiDB. The column `C` is simply the empty string `''` but not NULL.

3.6.3.3.5 `backslash-escape`

- Whether to interpret backslash escapes inside fields.
- If `backslash-escape` is true, the following sequences are recognized and transformed:

Sequence	Converted to
`\0`	Null character (U+0000)
`\b`	Backspace (U+0008)
`\n`	Line feed (U+000A)
`\r`	Carriage return (U+000D)
`\t`	Tab (U+0009)
`\Z`	Windows EOF (U+001A)

  In all other cases (for example, `\"`) the backslash is simply stripped, leaving the next character (`"`) in the field. The character left has no special roles (for example, delimiters) and is just an ordinary character.
- Quoting will not affect whether backslash escapes are interpreted.
- Corresponds to the `FIELDS ESCAPED BY '\'` option in the `LOAD DATA` statement.

3.6.3.3.6 `trim-last-separator`

- Treats the field `separator` as a terminator, and removes all trailing separators.

  For example, with the CSV file:

```
A,,B,,
```

- When `trim-last-separator = false`, this is interpreted as a row of 5 fields `('A', '', 'B', '', '')`.
- When `trim-last-separator = true`, this is interpreted as a row of 3 fields `('A', '', 'B')`.

3.6.3.3.7 Non-configurable options

TiDB Lightning does not support every option supported by the `LOAD DATA` statement. Some examples:

- The line terminator must only be CR (`\r`), LF (`\n`) or CRLF (`\r\n`), which means `LINES TERMINATED BY` is not customizable.
- There cannot be line prefixes (`LINES STARTING BY`).
- The header cannot be simply skipped (`IGNORE n LINES`), it must be valid column names if present.
- Delimiters and separators can only be a single ASCII character.

3.6.3.4 Common configurations

3.6.3.4.1 CSV

The default setting is already tuned for CSV following RFC 4180.

```
[mydumper.csv]
separator = ','
delimiter = '"'
header = true
not-null = false
null = '\N'
backslash-escape = true
trim-last-separator = false
```

Example content:

```
ID,Region,Count
1,"East",32
2,"South",\N
3,"West",10
4,"North",39
```

3.6.3.4.2 TSV

```
[mydumper.csv]
separator = "\t"
delimiter = ''
header = true
not-null = false
null = 'NULL'
backslash-escape = false
trim-last-separator = false
```

Example content:

```
ID    Region    Count
1     East      32
2     South     NULL
3     West      10
4     North     39
```

3.6.3.4.3 TPC-H DBGEN

```
[mydumper.csv]
separator = '|'
delimiter = ''
header = false
not-null = true
backslash-escape = false
trim-last-separator = true
```

Example content:

```
1|East|32|
2|South|0|
3|West|10|
4|North|39|
```

3.6.4 Migrate from MySQL SQL Files Using TiDB Lightning

This document describes how to migrate data from MySQL SQL files to TiDB using TiDB Lightning. For details on how to generate MySQL SQL files, refer to Mydumper or Dumpling.

The data migration process described in this document uses TiDB Lightning. The steps are as follows.

3.6.4.1 Step 1: Deploy TiDB Lightning

Before you start the migration, deploy TiDB Lightning.

Note:

- If you choose the Importer-backend to import data, you need to deploy `tikv-importer` along with TiDB Lightning. During the import process, the TiDB cluster cannot provide services. This mode imports data quickly, which is suitable for importing a large amount of data (above the TB level).
- If you choose the TiDB-backend, deploy TiDB Lightning only. The cluster can provide services normally during the import.
- For detailed differences between the two backend mode, see TiDB Lightning Backend.

3.6.4.2 Step 2: Configure data source of TiDB Lightning

This document takes the TiDB-backend as an example. Create the `tidb-lightning.toml` configuration file and add the following major configurations in the file:

1. Set the `data-source-dir` under `[mydumper]` to the path of the MySQL SQL file.

```
[mydumper]
# Data source directory
data-source-dir = "/data/export"
```

> **Note:**
> If a corresponding schema already exists in the downstream, set `no-schema=true` to skip the creation of the schema.

2. Add the configuration of the target TiDB cluster.

```
[tidb]
# The target cluster information. Fill in one address of tidb-server.
host = "172.16.31.1"
port = 4000
user = "root"
password = ""
```

For other configurations, see TiDB Lightning Configuration.

3.6.4.3 Step 3: Run TiDB Lightning to import data

Run TiDB Lightning to start the import operation. If you start TiDB Lightning by using `nohup` directly in the command line, the program might exit because of the `SIGHUP` signal. Therefore, it is recommended to write `nohup` in a script. For example:

```
### !/bin/bash
nohup ./tidb-lightning -config tidb-lightning.toml > nohup.out &
```

When the import operation is started, view the progress by the following two ways:

- `grep` the keyword `progress` in logs, which is updated every 5 minutes by default.
- Access the monitoring dashboard. See Monitor TiDB Lightning for details.

3.7 Maintain

3.7.1 TiDB Ansible Common Operations

This guide describes the common operations when you administer a TiDB cluster using TiDB Ansible.

3.7.1.1 Start a cluster

```
ansible-playbook start.yml
```

This operation starts all the components in the entire TiDB cluster in order, which include PD, TiDB, TiKV, and the monitoring components.

3.7.1.2 Stop a cluster

```
ansible-playbook stop.yml
```

This operation stops all the components in the entire TiDB cluster in order, which include PD, TiDB, TiKV, and the monitoring components.

3.7.1.3 Clean up cluster data

```
ansible-playbook unsafe_cleanup_data.yml
```

This operation stops the TiDB, Pump, TiKV and PD services, and cleans up the data directory of Pump, TiKV and PD.

3.7.1.4 Destroy a cluster

```
ansible-playbook unsafe_cleanup.yml
```

This operation stops the cluster and cleans up the data directory.

> **Note:**
>
> If the deployment directory is a mount point, an error will be reported, but implementation results remain unaffected, so you can ignore it.

3.7.2 Backup and Restore

3.7.2.1 Use Mydumper and TiDB Lightning for Data Backup and Restoration

This document describes how to perform full backup and restoration of the TiDB data using Mydumper and TiDB Lightning. For incremental backup and restoration, refer to TiDB Binlog.

Suppose that the TiDB service information is as follows:

Name	Address	Port	User	Password
TiDB	127.0.0.1	4000	root	*

Use the following tools for data backup and restoration:

- Mydumper: to export data from TiDB
- TiDB Lightning: to import data into TiDB

3.7.2.1.1 Full backup and restoration using Mydumper/TiDB Lightning

`mydumper` is a powerful data backup tool. For more information, refer to `maxbube/mydumper`.

Use Mydumper to export data from TiDB and use TiDB Lightning to import data into TiDB.

> **Note:**
>
> It is recommended to download Mydumper from the PingCAP website, because the R&D team has adapted `mydumper` for TiDB. It is not recommended to use `mysqldump` which is much slower for both backup and restoration.

Best practices for full backup and restoration using Mydumper/TiDB Lightning

To quickly backup and restore data (especially large amounts of data), refer to the following recommendations:

- Keep the exported data file as small as possible. It is recommended to use the `-F` parameter to set the file size. If you use TiDB Lightning to restore data, it is recommended that you set the value of `-F` to `256` (MB). If you use `loader` for restoration, it is recommended to set the value to `64` (MB).

3.7.2.1.2 Backup data from TiDB

Use `mydumper` to backup data from TiDB.

```
./bin/mydumper -h 127.0.0.1 -P 4000 -u root -t 32 -F 256 -B test -T t1,t2
    ↪ --skip-tz-utc -o ./var/test
```

In this command,

`-B test` means that the data is exported from the `test` database. `-T t1,t2` means that only the `t1` and `t2` tables are exported. `-t 32` means that 32 threads are used to export

the data. `-F 256` means that a table is partitioned into chunks, and one chunk is 256MB. `--skip-tz-utc` means to ignore the inconsistency of time zone setting between MySQL and the data exporting machine and to disable automatic conversion.

If `mydumper` returns the following error:

```
** (mydumper:27528): CRITICAL **: 13:25:09.081: Could not read data from
   ↪ testSchema.testTable: GC life time is shorter than transaction
   ↪ duration, transaction starts at 2019-08-05 21:10:01.451 +0800 CST, GC
   ↪  safe point is 2019-08-05 21:14:53.801 +0800 CST
```

Then execute two more commands:

1. Before executing the `mydumper` command, query the GC values of the TiDB cluster and adjust it to a suitable value using the MySQL client:

```
SELECT * FROM mysql.tidb WHERE VARIABLE_NAME = 'tikv_gc_life_time';
```

```
+-----------------------+------------------------------------------------------------
   ↪
| VARIABLE_NAME       | VARIABLE_VALUE
   ↪
   ↪ |
+-----------------------+------------------------------------------------------------
   ↪
| tikv_gc_life_time   | 10m0s
   ↪
   ↪ |
+-----------------------+------------------------------------------------------------
   ↪
1 rows in set (0.02 sec)
```

```
UPDATE mysql.tidb SET VARIABLE_VALUE = '720h' WHERE VARIABLE_NAME = '
   ↪ tikv_gc_life_time';
```

2. After running the `mydumper` command, adjust GC value of the TiDB cluster to its original value in step 1.

```
UPDATE mysql.tidb SET VARIABLE_VALUE = '10m' WHERE VARIABLE_NAME = '
   ↪ tikv_gc_life_time';
```

3.7.2.1.3 Restore data into TiDB

To restore data into TiDB, use TiDB Lightning to import the exported data. See TiDB Lightning Tutorial.

3.7.2.2 Export or Backup Data Using Dumpling

This document introduces how to use the Dumpling tool to export or backup data in TiDB. Dumpling exports data stored in TiDB as SQL or CSV data files and can be used to make a logical full backup or export.

For detailed usage of Dumpling, use the `--help` command or refer to Dumpling User Guide.

When using Dumpling, you need to execute the export command on a running cluster. This document assumes that there is a TiDB instance on the `127.0.0.1:4000` host and that this TiDB instance has a root user without a password.

3.7.2.2.1 Download Dumpling

To download the latest version of Dumpling, click the download link.

3.7.2.2.2 Export data from TiDB

Export to SQL files

Dumpling exports data to SQL files by default. You can also export data to SQL files by adding the `--filetype sql` flag:

```
dumpling \
  -u root \
  -P 4000 \
  -h 127.0.0.1 \
  --filetype sql \
  --threads 32 \
  -o /tmp/test \
  -F 256
```

In the above command, `-h`, `-P` and `-u` mean address, port and user, respectively. If password authentication is required, you can pass it to Dumpling with `-p $YOUR_SECRET_PASSWORD`.

Export to CSV files

If Dumpling exports data to CSV files (use `--filetype csv` to export to CSV files), you can also use `--sql <SQL>` to export the records selected by the specified SQL statement.

For example, you can export all records that match `id < 100` in `test.sbtest1` using the following command:

```
./dumpling \
  -u root \
  -P 4000 \
  -h 127.0.0.1 \
  -o /tmp/test \
```

```
  --filetype csv \
  --sql 'select * from `test`.`sbtest1` where id < 100'
```

Note:

- Currently, the `--sql` option can be used only for exporting to CSV files.
- Here you need to execute the `select * from <table-name> where id < 100` statement on all tables to be exported. If some tables do not have specified fields, the export fails.

Filter the exported data

Use the `--where` command to filter data

By default, Dumpling exports the tables of the entire database except the tables in the system databases. You can use `--where <SQL where expression>` to select the records to be exported.

```
./dumpling \
  -u root \
  -P 4000 \
  -h 127.0.0.1 \
  -o /tmp/test \
  --where "id < 100"
```

The above command exports the data that matches `id < 100` from each table.

Use the `--filter` command to filter data

Dumpling can filter specific databases or tables by specifying the table filter with the `--filter` command. The syntax of table filters is similar to that of `.gitignore`. For details, see Table Filter.

```
./dumpling \
  -u root \
  -P 4000 \
  -h 127.0.0.1 \
  -o /tmp/test \
  --filter "employees.*" \
  --filter "*.WorkOrder"
```

The above command exports all the tables in the `employees` database and the `WorkOrder` tables in all databases.

Use the `-B` or `-T` command to filter data

Dumpling can also export specific databases with the `-B` command or specific tables with the `-T` command.

> **Note:**
>
> - The `--filter` command and the `-T` command cannot be used at the same time.
> - The `-T` command can only accept a complete form of inputs like `database-name.table-name`, and inputs with only the table name are not accepted. Example: Dumpling cannot recognize `-T WorkOrder`.

Examples:

`-B employees` exports the `employees` database `-T employees.WorkOrder` exports the `employees.WorkOrder` table

Improve export efficiency through concurrency

The exported file is stored in the `./export-<current local time>` directory by default. Commonly used parameters are as follows:

- `-o` is used to select the directory where the exported files are stored.
- `-F` option is used to specify the maximum size of a single file (the unit here is `MiB`; inputs like `5GiB` or `8KB` are also acceptable).
- `-r` option is used to specify the maximum number of records (or the number of rows in the database) for a single file. When it is enabled, Dumpling enables concurrency in the table to improve the speed of exporting large tables.

You can use the above parameters to provide Dumpling with a higher degree of concurrency.

Adjust Dumpling's data consistency options

> **Note:**
>
> In most scenarios, you do not need to adjust the default data consistency options of Dumpling.

Dumpling uses the `--consistency <consistency level>` option to control the way in which data is exported for "consistency assurance". For TiDB, data consistency is guaranteed

by getting a snapshot of a certain timestamp by default (i.e. `--consistency snapshot`). When using snapshot for consistency, you can use the `--snapshot` parameter to specify the timestamp to be backed up. You can also use the following levels of consistency:

- `flush`: Use `FLUSH TABLES WITH READ LOCK` to ensure consistency.
- `snapshot`: Get a consistent snapshot of the specified timestamp and export it.
- `lock`: Add read locks on all tables to be exported.
- `none`: No guarantee for consistency.
- `auto`: Use `flush` for MySQL and `snapshot` for TiDB.

After everything is done, you can see the exported file in `/tmp/test`:

```
ls -lh /tmp/test | awk '{print $5 "\t" $9}'
```

```
140B metadata
66B  test-schema-create.sql
300B test.sbtest1-schema.sql
190K test.sbtest1.0.sql
300B test.sbtest2-schema.sql
190K test.sbtest2.0.sql
300B test.sbtest3-schema.sql
190K test.sbtest3.0.sql
```

In addition, if the data volume is very large, to avoid export failure due to GC during the export process, you can extend the GC time in advance:

```
update mysql.tidb set VARIABLE_VALUE = '720h' where VARIABLE_NAME = '
    ↪ tikv_gc_life_time';
```

After your operation is completed, set the GC time back (the default value is `10m`):

```
update mysql.tidb set VARIABLE_VALUE = '10m' where VARIABLE_NAME = '
    ↪ tikv_gc_life_time';
```

Finally, all the exported data can be imported back to TiDB using Lightning.

3.7.3 Identify Abnormal Queries

3.7.3.1 Identify Slow Queries

To help users identify slow queries, analyze and improve the performance of SQL execution, TiDB outputs the statements whose execution time exceeds slow-threshold (The default value is 300 milliseconds) to slow-query-file (The default value is "tidb-slow.log").

3.7.3.1.1 Usage example

```
#### Time: 2019-08-14T09:26:59.487776265+08:00
#### Txn_start_ts: 410450924122144769
#### User: root@127.0.0.1
#### Conn_ID: 3086
#### Query_time: 1.527627037
#### Parse_time: 0.000054933
#### Compile_time: 0.000129729
#### Process_time: 0.07 Request_count: 1 Total_keys: 131073 Process_keys:
    ↪ 131072 Prewrite_time: 0.335415029 Commit_time: 0.032175429
    ↪ Get_commit_ts_time: 0.000177098 Local_latch_wait_time: 0.106869448
    ↪ Write_keys: 131072 Write_size: 3538944 Prewrite_region: 1
#### DB: test
#### Is_internal: false
#### Digest: 50
    ↪ a2e32d2abbd6c1764b1b7f2058d428ef2712b029282b776beb9506a365c0f1
#### Stats: t:pseudo
#### Num_cop_tasks: 1
#### Cop_proc_avg: 0.07 Cop_proc_p90: 0.07 Cop_proc_max: 0.07 Cop_proc_addr:
    ↪  172.16.5.87:20171
#### Cop_wait_avg: 0 Cop_wait_p90: 0 Cop_wait_max: 0 Cop_wait_addr:
    ↪ 172.16.5.87:20171
#### Mem_max: 525211
#### Succ: true
#### Plan: tidb_decode_plan('
    ↪ ZJAwCTMyXzcJMAkyMAlkYXRhOlRhYmxlU2Nhbl82CjEJMTBfNgkxAR0AdAEY1Dp0LCByYW5nZTpbLWluZi
    ↪ ==')
insert into t select * from t;
```

3.7.3.1.2 Fields description

> **Note:**
>
> The unit of all the following time fields in the slow query log is **"second"**.

Slow query basics:

- `Time`: The print time of log.
- `Query_time`: The execution time of a statement.
- `Parse_time`: The parsing time for the statement.

- `Compile_time`: The duration of the query optimization.
- `Query`: A SQL statement. `Query` is not printed in the slow log, but the corresponding field is called `Query` after the slow log is mapped to the memory table.
- `Digest`: The fingerprint of the SQL statement.
- `Txn_start_ts`: The start timestamp and the unique ID of a transaction. You can use this value to search for the transaction-related logs.
- `Is_internal`: Whether a SQL statement is TiDB internal. `true` indicates that a SQL statement is executed internally in TiDB and `false` indicates that a SQL statement is executed by the user.
- `Index_ids`: The IDs of the indexes involved in a statement.
- `Succ`: Whether a statement is executed successfully.
- `Backoff_time`: The waiting time before retry when a statement encounters errors that require a retry. The common errors as such include: `lock occurs`, `Region split`, and `tikv server is busy`.
- `Plan`: The execution plan of the statement. Use the `select tidb_decode_plan('xxx ...')` statement to parse the specific execution plan.

The following fields are related to transaction execution:

- `Prewrite_time`: The duration of the first phase (prewrite) of the two-phase transaction commit.
- `Commit_time`: The duration of the second phase (commit) of the two-phase transaction commit.
- `Get_commit_ts_time`: The time spent on getting `commit_ts` during the second phase (commit) of the two-phase transaction commit.
- `Local_latch_wait_time`: The time that TiDB spends on waiting for the lock before the second phase (commit) of the two-phase transaction commit.
- `Write_keys`: The count of keys that the transaction writes to the Write CF in TiKV.
- `Write_size`: The total size of the keys or values to be written when the transaction commits.
- `Prewrite_region`: The number of TiKV Regions involved in the first phase (prewrite) of the two-phase transaction commit. Each Region triggers a remote procedure call.

Memory usage fields:

- `Mem_max`: The maximum memory space used during the execution period of a SQL statement (the unit is byte).

User fields:

- `User`: The name of the user who executes this statement.
- `Conn_ID`: The Connection ID (session ID). For example, you can use the keyword `con:3` to search for the log whose session ID is `3`.

- `DB`: The current database.

TiKV Coprocessor Task fields:

- `Request_count`: The number of Coprocessor requests that a statement sends.
- `Total_keys`: The number of keys that Coprocessor has scanned.
- `Process_time`: The total processing time of a SQL statement in TiKV. Because data is sent to TiKV concurrently, this value might exceed `Query_time`.
- `Wait_time`: The total waiting time of a statement in TiKV. Because the Coprocessor of TiKV runs a limited number of threads, requests might queue up when all threads of Coprocessor are working. When a request in the queue takes a long time to process, the waiting time of the subsequent requests increases.
- `Process_keys`: The number of keys that Coprocessor has processed. Compared with `total_keys`, `processed_keys` does not include the old versions of MVCC. A great difference between `processed_keys` and `total_keys` indicates that many old versions exist.
- `Cop_proc_avg`: The average execution time of cop-tasks.
- `Cop_proc_p90`: The P90 execution time of cop-tasks.
- `Cop_proc_max`: The maximum execution time of cop-tasks.
- `Cop_proc_addr`: The address of the cop-task with the longest execution time.
- `Cop_wait_avg`: The average waiting time of cop-tasks.
- `Cop_wait_p90`: The P90 waiting time of cop-tasks.
- `Cop_wait_max`: The maximum waiting time of cop-tasks.
- `Cop_wait_addr`: The address of the cop-task whose waiting time is the longest.

3.7.3.1.3 Memory mapping in slow log

You can query the contents of the slow query log by querying the `INFORMATION_SCHEMA.SLOW_QUERY` table. Each column name in the table corresponds to one field name in the slow log. For table structure, see the introduction to the `SLOW_QUERY` table in Information Schema.

> **Note:**
>
> Every time you query the `SLOW_QUERY` table, TiDB reads and parses the current slow query log.

3.7.3.1.4 Query example of SLOW_QUERY

Top-N slow queries

Query the Top 2 slow queries of users. `Is_internal=false` means excluding slow queries inside TiDB and only querying slow queries of users.

```
select query_time, query
from information_schema.slow_query
where is_internal = false
order by query_time desc
limit 2;
```

Usage example:

```
+-------------+-----------------------------------------------------------------------+
   ↪
| query_time | query                                                                  |
+-------------+-----------------------------------------------------------------------+
   ↪
| 12.77583857 | select * from t_slim, t_wide where t_slim.c0=t_wide.c0; |
|  0.734982725 | select t0.c0, t1.c1 from t_slim t0, t_wide t1 where t0.c0=
   ↪ t1.c0; |
+-------------+-----------------------------------------------------------------------+
   ↪
```

Query the Top-N slow queries of the `test` user

In the following example, the slow queries executed by the `test` user are queried, and the first two results are displayed in reverse order of execution time.

```
select query_time, query, user
from information_schema.slow_query
where is_internal = false
  and user = "test"
order by query_time desc
limit 2;
```

Usage example:

```
+------------+-------------------------------------------------------------------+------
   ↪
| Query_time | query                                                              |
   ↪ user         |
+------------+-------------------------------------------------------------------+------
   ↪
| 0.676408014 | select t0.c0, t1.c1 from t_slim t0, t_wide t1 where t0.c0=t1
   ↪ .c1; | test |
+------------+-------------------------------------------------------------------+------
   ↪
```

Query similar slow queries with the same SQL fingerprints

After querying the Top-N SQL statements, continue to query similar slow queries using the same fingerprints.

1. Acquire Top-N slow queries and the corresponding SQL fingerprints.

```
select query_time, query, digest
from information_schema.slow_query
where is_internal = false
order by query_time desc
limit 1;
```

Usage example:

```
+-------------+-----------------------------+-----------------------------------
   ↪
| query_time | query                       | digest
   ↪                                                          |
+-------------+-----------------------------+-----------------------------------
   ↪
| 0.302558006 | select * from t1 where a=1; | 4751
   ↪ cb6008fda383e22dacb601fde85425dc8f8cf669338d55d944bafb46a6fa |
+-------------+-----------------------------+-----------------------------------
   ↪
```

2. Query similar slow queries with the fingerprints.

```
select query, query_time
from information_schema.slow_query
where digest = "4751
   ↪ cb6008fda383e22dacb601fde85425dc8f8cf669338d55d944bafb46a6fa";
```

Usage example:

```
+-----------------------------+-------------+
| query                       | query_time |
+-----------------------------+-------------+
| select * from t1 where a=1; | 0.302558006 |
| select * from t1 where a=2; | 0.401313532 |
+-----------------------------+-------------+
```

3.7.3.1.5 Query slow queries with pseudo stats

```
select query, query_time, stats
from information_schema.slow_query
where is_internal = false
  and stats like '%pseudo%';
```

Usage example:

```
+----------------------------+------------+--------------------------------+
    ↪
| query                      | query_time | stats                          |
+----------------------------+------------+--------------------------------+
    ↪
| select * from t1 where a=1; | 0.302558006 | t1:pseudo                     |
| select * from t1 where a=2; | 0.401313532 | t1:pseudo                     |
| select * from t1 where a>2; | 0.602011247 | t1:pseudo                     |
| select * from t1 where a>3; | 0.50077719 | t1:pseudo                      |
| select * from t1 join t2; | 0.931260518 | t1:407872303825682445,t2:pseudo
    ↪ |
+----------------------------+------------+--------------------------------+
    ↪
```

Parse other TiDB slow log files

TiDB uses the session variable `tidb_slow_query_file` to control the files to be read and parsed when querying `INFORMATION_SCHEMA.SLOW_QUERY`. You can query the contents of other slow query log files by modifying the value of the session variable.

```
set tidb_slow_query_file = "/path-to-log/tidb-slow.log"
```

Parse TiDB slow logs with `pt-query-digest`

Use `pt-query-digest` to parse TiDB slow logs.

> **Note:**
>
> It is recommended to use `pt-query-digest` 3.0.13 or later versions.

For example:

```
pt-query-digest --report tidb-slow.log
```

Usage example:

```
#### 320ms user time, 20ms system time, 27.00M rss, 221.32M vsz
#### Current date: Mon Mar 18 13:18:51 2019
#### Hostname: localhost.localdomain
#### Files: tidb-slow.log
#### Overall: 1.02k total, 21 unique, 0 QPS, 0x concurrency
    ↪ ______________________
#### Time range: 2019-03-18-12:22:16 to 2019-03-18-13:08:52
#### Attribute       total    min    max    avg    95%  stddev median
```

```
#### ============  ======= ======= ======= ======= ======= ======= =======
#### Exec time        218s    10ms    13s   213ms    30ms     1s    19ms
#### Query size     175.37k     9   2.01k 175.89 158.58 122.36 158.58
#### Commit time      46ms    2ms    7ms    3ms    7ms    1ms    3ms
#### Conn ID           71      1     16   8.88  15.25   4.06   9.83
#### Process keys   581.87k     2 103.15k 596.43 400.73  3.91k 400.73
#### Process time      31s    1ms    10s   32ms   19ms  334ms   16ms
#### Request coun    1.97k      1     10   2.02   1.96   0.33   1.96
#### Total keys     636.43k     2 103.16k 652.35 793.42  3.97k 400.73
#### Txn start ts   374.38E     0  16.00E 375.48P 1.25P 89.05T  1.25P
#### Wait time        943ms    1ms   19ms    1ms    2ms    1ms  972us
.
.
.
```

3.7.3.1.6 Identify problematic SQL statements

Not all of the `SLOW_QUERY` statements are problematic. Only those whose `process_time` is very large increase the pressure on the entire cluster.

The statements whose `wait_time` is very large and `process_time` is very small are usually not problematic. This is because the statement is blocked by real problematic statements and it has to wait in the execution queue, which leads to a much longer response time.

`admin show slow` command

In addition to the TiDB log file, you can identify slow queries by running the `admin show slow` command:

```
admin show slow recent N
admin show slow top [internal | all] N
```

`recent N` shows the recent N slow query records, for example:

```
admin show slow recent 10
```

`top N` shows the slowest N query records recently (within a few days). If the `internal` option is provided, the returned results would be the inner SQL executed by the system; If the `all` option is provided, the returned results would be the user's SQL combinated with inner SQL; Otherwise, this command would only return the slow query records from the user's SQL.

```
admin show slow top 3
admin show slow top internal 3
admin show slow top all 5
```

TiDB stores only a limited number of slow query records because of the limited memory. If the value of `N` in the query command is greater than the records count, the number of

returned records is smaller than `N`.

The following table shows output details:

Column name	Description
start	The starting time of the SQL execution
duration	The duration of the SQL execution
details	The details of the SQL execution

Column name	Description
succ	Whether the SQL statement is executed successfully. `1` means success and `0` means failure.
conn_id	The connection ID for the session
transcation_ts	The `commit ↪ ↪ ts` for a transaction commit

Column name	Description
user	The user name for the execution of the statement
db	The database involved when the statement is executed
table_ids	The ID of the table involved when the SQL statement is executed

Column name	Description
index_ids	The ID of the index involved when the SQL statement is executed
internal	This is a TiDB internal SQL statement
digest	The fingerprint of the SQL statement
sql	The SQL statement that is being executed or has been executed

3.7.3.2 Identify Expensive Queries

TiDB allows you to identify expensive queries during SQL execution, so you can diagnose and improve the performance of SQL execution. Specifically, TiDB prints the information about statements whose execution time exceeds `tidb_expensive_query_time_threshold` (60 seconds by default) or memory usage exceeds `mem-quota-query` (32 GB by default) to the tidb-server log file ("tidb.log" by default).

> **Note:**
>
> The expensive query log differs from the slow query log in this way: TiDB prints statement information to the expensive query log **as soon as** the statement exceeds the threshold of resource usage (execution time or memory usage); while TiDB prints statement information to the slow query log **after** the statement execution.

3.7.3.2.1 Expensive query log example

```
[2020/02/05 15:32:25.096 +08:00] [WARN] [expensivequery.go:167] [
   ↪ expensive_query] [cost_time=60.008338935s] [wait_time=0s] [
   ↪ request_count=1] [total_keys=70] [process_keys=65] [num_cop_tasks=1]
   ↪ [process_avg_time=0s] [process_p90_time=0s] [process_max_time=0s] [
   ↪ process_max_addr=10.0.1.9:20160] [wait_avg_time=0.002s] [
   ↪ wait_p90_time=0.002s] [wait_max_time=0.002s] [wait_max_addr
   ↪ =10.0.1.9:20160] [stats=t:pseudo] [conn_id=60026] [user=root] [
   ↪ database=test] [table_ids="[122]"] [txn_start_ts=414420273735139329]
   ↪ [mem_max="1035 Bytes (1.0107421875 KB)"] [sql="insert into t select
   ↪ sleep(1) from t"]
```

3.7.3.2.2 Fields description

Basic fields:

- `cost_time`: The execution time of a statement when the log is printed.
- `stats`: The version of statistics used by the tables or indexes involved in a statement. If the value is `pseudo`, it means that there are no available statistics. In this case, you need to analyze the tables or indexes.
- `table_ids`: The IDs of the tables involved in a statement.
- `txn_start_ts`: The start timestamp and the unique ID of a transaction. You can use this value to search for the transaction-related logs.
- `sql`: The sql statement.

Memory usage related fields:

- `mem_max`: Memory usage of a statement when the log is printed. This field has two kinds of units to measure memory usage: byte and other readable and adaptable units (such as MB and GB).

User related fields:

- `user`: The name of the user who executes the statement.
- `conn_id`: The connection ID (session ID). For example, you can use the keyword `con:60026` to search for the log whose session ID is `60026`.
- `database`: The database where the statement is executed.

TiKV Coprocessor task related fields:

- `wait_time`: The total waiting time of all Coprocessor requests of a statement in TiKV. Because the Coprocessor of TiKV runs a limited number of threads, requests might queue up when all threads of Coprocessor are working. When a request in the queue takes a long time to process, the waiting time of the subsequent requests increases.
- `request_count`: The number of Coprocessor requests that a statement sends.
- `total_keys`: The number of keys that Coprocessor has scanned.
- `processed_keys`: The number of keys that Coprocessor has processed. Compared with `total_keys`, `processed_keys` does not include the old versions of MVCC. A great difference between `processed_keys` and `total_keys` indicates that many old versions exist.
- `num_cop_tasks`: The number of Coprocessor requests that a statement sends.
- `process_avg_time`: The average execution time of Coprocessor tasks.
- `process_p90_time`: The P90 execution time of Coprocessor tasks.
- `process_max_time`: The maximum execution time of Coprocessor tasks.
- `process_max_addr`: The address of the Coprocessor task with the longest execution time.
- `wait_avg_time`: The average waiting time of Coprocessor tasks.
- `wait_p90_time`: The P90 waiting time of Coprocessor tasks.
- `wait_max_time`: The maximum waiting time of Coprocessor tasks.
- `wait_max_addr`: The address of the Coprocessor task with the longest waiting time.

3.8 Scale

3.8.1 Scale the TiDB Cluster Using TiDB Ansible

The capacity of a TiDB cluster can be increased or decreased without affecting the online services.

Warning:

In decreasing the capacity, if your cluster has a mixed deployment of other services, do not perform the following procedures. The following examples assume that the removed nodes have no mixed deployment of other services.

Assume that the topology is as follows:

Name	Host IP	Services
node1	172.16.10.1	PD1
node2	172.16.10.2	PD2
node3	172.16.10.3	PD3, Monitor
node4	172.16.10.4	TiDB1
node5	172.16.10.5	TiDB2
node6	172.16.10.6	TiKV1
node7	172.16.10.7	TiKV2
node8	172.16.10.8	TiKV3
node9	172.16.10.9	TiKV4

3.8.1.1 Increase the capacity of a TiDB/TiKV node

For example, if you want to add two TiDB nodes (node101, node102) with the IP addresses `172.16.10.101` and `172.16.10.102`, take the following steps:

1. Edit the `inventory.ini` file and the `hosts.ini` file, and append the node information.

    - Edit the `inventory.ini` file:

```
[tidb_servers]
172.16.10.4
172.16.10.5
172.16.10.101
172.16.10.102

[pd_servers]
172.16.10.1
172.16.10.2
172.16.10.3

[tikv_servers]
172.16.10.6
172.16.10.7
172.16.10.8
```

```
172.16.10.9

[monitored_servers]
172.16.10.1
172.16.10.2
172.16.10.3
172.16.10.4
172.16.10.5
172.16.10.6
172.16.10.7
172.16.10.8
172.16.10.9
172.16.10.101
172.16.10.102

[monitoring_servers]
172.16.10.3

[grafana_servers]
172.16.10.3
```

Now the topology is as follows:

Name	Host IP	Services
node1	172.16.10.1	PD1
node2	172.16.10.2	PD2
node3	172.16.10.3	PD3, Monitor
node4	172.16.10.4	TiDB1
node5	172.16.10.5	TiDB2
node101	**172.16.10.101**	**TiDB3**
node102	**172.16.10.102**	**TiDB4**
node6	172.16.10.6	TiKV1
node7	172.16.10.7	TiKV2
node8	172.16.10.8	TiKV3
node9	172.16.10.9	TiKV4

- Edit the `hosts.ini` file:

```
[servers]
172.16.10.1
172.16.10.2
172.16.10.3
172.16.10.4
172.16.10.5
172.16.10.6
```

```
172.16.10.7
172.16.10.8
172.16.10.9
172.16.10.101
172.16.10.102
[all:vars]
username = tidb
ntp_server = pool.ntp.org
```

2. Initialize the newly added node.

    1. Configure the SSH mutual trust and sudo rules of the deployment machine on the central control machine:

    ```
    ansible-playbook -i hosts.ini create_users.yml -l
       ↪ 172.16.10.101,172.16.10.102 -u root -k
    ```

    2. Install the NTP service on the deployment target machine:

    ```
    ansible-playbook -i hosts.ini deploy_ntp.yml -u tidb -b
    ```

    3. Initialize the node on the deployment target machine:

    ```
    ansible-playbook bootstrap.yml -l 172.16.10.101,172.16.10.102
    ```

> **Note:**
>
> If an alias is configured in the `inventory.ini` file, for example, `node101 ↪ ansible_host=172.16.10.101`, use `-l` to specify the alias when executing `ansible-playbook`. For example, `ansible-playbook bootstrap ↪ .yml -l node101,node102`. This also applies to the following steps.

3. Deploy the newly added node:

```
ansible-playbook deploy.yml -l 172.16.10.101,172.16.10.102
```

4. Start the newly added node:

```
ansible-playbook start.yml -l 172.16.10.101,172.16.10.102
```

5. Update the Prometheus configuration and restart the cluster:

```
ansible-playbook rolling_update_monitor.yml --tags=prometheus
```

6. Monitor the status of the entire cluster and the newly added node by opening a browser to access the monitoring platform: `http://172.16.10.3:3000`.

You can use the same procedure to add a TiKV node. But to add a PD node, some configuration files need to be manually updated.

3.8.1.2 Increase the capacity of a PD node

For example, if you want to add a PD node (node103) with the IP address `172.16.10.103` ↪ , take the following steps:

1. Edit the `inventory.ini` file and append the node information to the end of the `[` ↪ `pd_servers]` group:

```
[tidb_servers]
172.16.10.4
172.16.10.5

[pd_servers]
172.16.10.1
172.16.10.2
172.16.10.3
172.16.10.103

[tikv_servers]
172.16.10.6
172.16.10.7
172.16.10.8
172.16.10.9

[monitored_servers]
172.16.10.4
172.16.10.5
172.16.10.1
172.16.10.2
172.16.10.3
172.16.10.103
172.16.10.6
172.16.10.7
172.16.10.8
172.16.10.9

[monitoring_servers]
172.16.10.3

[grafana_servers]
172.16.10.3
```

Now the topology is as follows:

Name	Host IP	Services
node1	172.16.10.1	PD1

Name	Host IP	Services
node2	172.16.10.2	PD2
node3	172.16.10.3	PD3, Monitor
node103	**172.16.10.103**	**PD4**
node4	172.16.10.4	TiDB1
node5	172.16.10.5	TiDB2
node6	172.16.10.6	TiKV1
node7	172.16.10.7	TiKV2
node8	172.16.10.8	TiKV3
node9	172.16.10.9	TiKV4

2. Initialize the newly added node:

```
ansible-playbook bootstrap.yml -l 172.16.10.103
```

3. Deploy the newly added node:

```
ansible-playbook deploy.yml -l 172.16.10.103
```

4. Login the newly added PD node and edit the starting script:

```
{deploy_dir}/scripts/run_pd.sh
```

1. Remove the `--initial-cluster="xxxx" \` configuration.

> **Note:**
> You cannot add the `#` character at the beginning of the line. Otherwise, the following configuration cannot take effect.

2. Add `--join="http://172.16.10.1:2379" \`. The IP address (`172.16.10.1`) can be any of the existing PD IP address in the cluster.
3. Start the PD service in the newly added PD node:

```
{deploy_dir}/scripts/start_pd.sh
```

> **Note:**
> Before start, you need to ensure that the `health` status of the newly added PD node is "true", using PD Control. Otherwise, the PD service might fail to start and an error message `["join meet error ↪ "] [error="etcdserver: unhealthy cluster"]` is returned in the log.

4. Use `pd-ctl` to check whether the new node is added successfully:

```
./pd-ctl -u "http://172.16.10.1:2379"
```

> **Note:**
> `pd-ctl` is a command used to check the number of PD nodes.

5. Start the monitoring service:

```
ansible-playbook start.yml -l 172.16.10.103
```

> **Note:**
> If you use an alias (inventory_name), use the `-l` option to specify the alias.

6. Update the cluster configuration:

```
ansible-playbook deploy.yml
```

7. Restart Prometheus, and enable the monitoring of PD nodes used for increasing the capacity:

```
ansible-playbook stop.yml --tags=prometheus
ansible-playbook start.yml --tags=prometheus
```

8. Monitor the status of the entire cluster and the newly added node by opening a browser to access the monitoring platform: `http://172.16.10.3:3000`.

> **Note:**
>
> The PD Client in TiKV caches the list of PD nodes. Currently, the list is updated only if the PD leader is switched or the TiKV server is restarted to load the latest configuration. To avoid TiKV caching an outdated list, there should be at least two existing PD members in the PD cluster after increasing or decreasing the capacity of a PD node. If this condition is not met, transfer the PD leader manually to update the list of PD nodes.

3.8.1.3 Decrease the capacity of a TiDB node

For example, if you want to remove a TiDB node (node5) with the IP address `172.16.10.5`, take the following steps:

1. Stop all services on node5:

```
ansible-playbook stop.yml -l 172.16.10.5
```

2. Edit the `inventory.ini` file and remove the node information:

```
[tidb_servers]
172.16.10.4
#172.16.10.5 # the removed node

[pd_servers]
172.16.10.1
172.16.10.2
172.16.10.3

[tikv_servers]
172.16.10.6
172.16.10.7
172.16.10.8
172.16.10.9

[monitored_servers]
172.16.10.4
#172.16.10.5 # the removed node
172.16.10.1
172.16.10.2
172.16.10.3
172.16.10.6
172.16.10.7
172.16.10.8
172.16.10.9

[monitoring_servers]
172.16.10.3

[grafana_servers]
172.16.10.3
```

Now the topology is as follows:

Name	Host IP	Services
node1	172.16.10.1	PD1
node2	172.16.10.2	PD2
node3	172.16.10.3	PD3, Monitor
node4	172.16.10.4	TiDB1
node5	**172.16.10.5**	**TiDB2 removed**
node6	172.16.10.6	TiKV1
node7	172.16.10.7	TiKV2
node8	172.16.10.8	TiKV3
node9	172.16.10.9	TiKV4

3. Update the Prometheus configuration and restart the cluster:

```
ansible-playbook rolling_update_monitor.yml --tags=prometheus
```

4. Monitor the status of the entire cluster by opening a browser to access the monitoring platform: `http://172.16.10.3:3000`.

3.8.1.4 Decrease the capacity of a TiKV node

For example, if you want to remove a TiKV node (node9) with the IP address `172.16.10.9`, take the following steps:

1. Remove the node from the cluster using `pd-ctl`:

    1. View the store ID of node9:

    ```
    ./pd-ctl -u "http://172.16.10.1:2379" -d store
    ```

    2. Remove node9 from the cluster, assuming that the store ID is 10:

    ```
    ./pd-ctl -u "http://172.16.10.1:2379" -d store delete 10
    ```

2. Use `pd-ctl` to check whether the node is successfully removed:

```
./pd-ctl -u "http://172.16.10.1:2379" -d store 10
```

> **Note:**
> It takes some time to remove the node. If the status of the node you remove becomes Tombstone, then this node is successfully removed.

3. After the node is successfully removed, stop the services on node9:

```
ansible-playbook stop.yml -l 172.16.10.9
```

4. Edit the `inventory.ini` file and remove the node information:

```
[tidb_servers]
172.16.10.4
172.16.10.5

[pd_servers]
172.16.10.1
172.16.10.2
172.16.10.3

[tikv_servers]
172.16.10.6
172.16.10.7
172.16.10.8
#172.16.10.9 # the removed node

[monitored_servers]
172.16.10.4
172.16.10.5
172.16.10.1
172.16.10.2
172.16.10.3
172.16.10.6
172.16.10.7
172.16.10.8
#172.16.10.9 # the removed node

[monitoring_servers]
172.16.10.3

[grafana_servers]
172.16.10.3
```

Now the topology is as follows:

Name	Host IP	Services
node1	172.16.10.1	PD1
node2	172.16.10.2	PD2
node3	172.16.10.3	PD3, Monitor
node4	172.16.10.4	TiDB1
node5	172.16.10.5	TiDB2

Name	Host IP	Services
node6	172.16.10.6	TiKV1
node7	172.16.10.7	TiKV2
node8	172.16.10.8	TiKV3
node9	**172.16.10.9**	**TiKV4 removed**

5. Update the Prometheus configuration and restart the cluster:

```
ansible-playbook rolling_update_monitor.yml --tags=prometheus
```

6. Monitor the status of the entire cluster by opening a browser to access the monitoring platform: `http://172.16.10.3:3000`.

3.8.1.5 Decrease the capacity of a PD node

For example, if you want to remove a PD node (node2) with the IP address `172.16.10.2`, take the following steps:

1. Remove the node from the cluster using `pd-ctl`:

    1. View the name of node2:

    ```
    ./pd-ctl -u "http://172.16.10.1:2379" -d member
    ```

    2. Remove node2 from the cluster, assuming that the name is pd2:

    ```
    ./pd-ctl -u "http://172.16.10.1:2379" -d member delete name pd2
    ```

2. Use Grafana or `pd-ctl` to check whether the node is successfully removed:

```
./pd-ctl -u "http://172.16.10.1:2379" -d member
```

3. After the node is successfully removed, stop the services on node2:

```
ansible-playbook stop.yml -l 172.16.10.2
```

> **Note:**
> In this example, you can only stop the PD service on node2. If there are any other services deployed with the IP address `172.16.10.2`, use the `-t` option to specify the service (such as `-t tidb`).

4. Edit the `inventory.ini` file and remove the node information:

```
[tidb_servers]
172.16.10.4
172.16.10.5

[pd_servers]
172.16.10.1
#172.16.10.2 # the removed node
172.16.10.3

[tikv_servers]
172.16.10.6
172.16.10.7
172.16.10.8
172.16.10.9

[monitored_servers]
172.16.10.4
172.16.10.5
172.16.10.1
#172.16.10.2 # the removed node
172.16.10.3
172.16.10.6
172.16.10.7
172.16.10.8
172.16.10.9

[monitoring_servers]
172.16.10.3

[grafana_servers]
172.16.10.3
```

Now the topology is as follows:

Name	Host IP	Services
node1	172.16.10.1	PD1
node2	**172.16.10.2**	**PD2 removed**
node3	172.16.10.3	PD3, Monitor
node4	172.16.10.4	TiDB1
node5	172.16.10.5	TiDB2
node6	172.16.10.6	TiKV1
node7	172.16.10.7	TiKV2
node8	172.16.10.8	TiKV3
node9	172.16.10.9	TiKV4

Name	Host IP	Services

5. Update the cluster configuration:

```
ansible-playbook deploy.yml
```

6. Restart Prometheus, and disable the monitoring of PD nodes used for increasing the capacity:

```
ansible-playbook stop.yml --tags=prometheus
ansible-playbook start.yml --tags=prometheus
```

7. To monitor the status of the entire cluster, open a browser to access the monitoring platform: `http://172.16.10.3:3000`.

> **Note:**
>
> The PD Client in TiKV caches the list of PD nodes. Currently, the list is updated only if the PD leader is switched or the TiKV server is restarted to load the latest configuration. To avoid TiKV caching an outdated list, there should be at least two existing PD members in the PD cluster after increasing or decreasing the capacity of a PD node. If this condition is not met, transfer the PD leader manually to update the list of PD nodes.

3.8.2 Scale a TiDB cluster

3.8.2.1 Overview

The capacity of a TiDB cluster can be increased or reduced without affecting online services.

> **Note:**
>
> If your TiDB cluster is deployed using TiDB Ansible, see Scale the TiDB Cluster Using TiDB Ansible.

The following part shows you how to add or delete PD, TiKV or TiDB nodes.

About `pd-ctl` usage, refer to PD Control User Guide.

3.8.2.2 PD

Assume we have three PD servers with the following details:

Name	ClientUrls	PeerUrls
pd1	http://host1:2379	http://host1:2380
pd2	http://host2:2379	http://host2:2380
pd3	http://host3:2379	http://host3:2380

Get the information about the existing PD nodes through pd-ctl:

```
./pd-ctl -u http://host1:2379 -i
```

```
>> member
```

3.8.2.2.1 Add a node dynamically

Add a new PD server to the current PD cluster by using the parameter `join`. To add `pd4`, you just need to specify the client url of any PD server in the PD cluster in the parameter `--join`, like:

```
./bin/pd-server --name=pd4 \
                --client-urls="http://host4:2379" \
                --peer-urls="http://host4:2380" \
                --join="http://host1:2379"
```

3.8.2.2.2 Delete a node dynamically

Delete `pd4` through pd-ctl:

```
./pd-ctl -u http://host1:2379
```

```
>> member delete name pd4
```

3.8.2.2.3 Migrate a node dynamically

If you want to migrate a node to a new machine, you need to, first of all, add a node on the new machine and then delete the node on the old machine. As you can just migrate one node at a time, if you want to migrate multiple nodes, you need to repeat the above steps until you have migrated all nodes. After completing each step, you can verify the process by checking the information of all nodes.

3.8.2.3 TiKV

Get the information about the existing TiKV nodes through pd-ctl:

```
./pd-ctl -u http://host1:2379
```

```
>> store
```

3.8.2.3.1 Add a node dynamically

It is very easy to add a new TiKV server dynamically. You just need to start a TiKV server on the new machine. The newly started TiKV server will automatically register in the existing PD of the cluster. To reduce the pressure of the existing TiKV servers, PD loads balance automatically, which means PD gradually migrates some data to the new TiKV server.

3.8.2.3.2 Delete a node dynamically

To delete (make it offline) a TiKV server safely, you need to inform PD in advance. After that, PD is able to migrate the data on this TiKV server to other TiKV servers, ensuring that data have enough replicas.

Assume that you need to delete the TiKV server with a store id 1, you can complete this through pd-ctl:

```
./pd-ctl -u http://host1:2379
```

```
>> store delete 1
```

Then you can check the state of this TiKV:

```
./pd-ctl -u http://host1:2379
```

```
>> store 1
```

```
{
  "store": {
    "id": 1,
    "address": "127.0.0.1:21060",
    "state": 1,
    "state_name": "Offline"
  },
  "status": {
    ...
  }
}
```

You can verify the state of this store using `state_name`:

- Up: This store is in service.
- Disconnected: The heartbeats of this store cannot be detected currently, which might be caused by a failure or network interruption.
- Down: PD does not receive heartbeats from the TiKV store for more than an hour (the time can be configured using `max-down-time`). At this time, PD adds a replica for the data on this store.
- Offline: The store is in the process of transferring its Regions to other nodes. The state name is misleading: the store is available and even continuing to lead some of its Regions.
- Tombstone: This store is shut down and has no data on it, so the instance can be deleted.

3.8.2.3.3 Migrate a node dynamically

To migrate TiKV servers to a new machine, you also need to add nodes on the new machine and then make all nodes on the old machine offline. In the process of migration, you can add all machines in the new cluster to the existing cluster, then make old nodes offline one by one. To verify whether a node has been made offline, you can check the state information of the node in process. After verifying, you can make the next node offline.

3.8.2.4 TiDB

TiDB is a stateless server, which means it can be added or deleted directly. It should be noted that if you deploy a proxy (such as HAProxy) in front of TiDB, you need to update the proxy configuration and reload it.

3.9 Upgrade

3.9.1 TiDB 3.0 Upgrade Guide

This document is targeted for users who want to upgrade from TiDB 2.0 or 2.1 to 3.0 versions, or from an earlier 3.0 version to later 3.0 versions. TiDB 3.0 is compatible with TiDB Binlog of the cluster version.

3.9.1.1 Upgrade caveat

- Rolling back to 2.1.x or earlier versions after upgrading is not supported.
- Before upgrading to 3.0 from 2.0.6 or earlier versions, check if there are any running DDL operations, especially time-consuming ones like `Add Index`. If there are any, wait for the DDL operations to finish before you upgrade.

- Parallel DDL is supported in TiDB 2.1 and later versions. Therefore, for clusters with a TiDB version earlier than 2.0.1, rolling update to TiDB 3.0 is not supported. To upgrade, you can choose either of the following two options:
  - Stop the cluster and upgrade to 3.0 directly.
  - Roll update to 2.0.1 or later 2.0.x versions, and then roll update to the 3.0 version.

> **Note:**
>
> Do not execute any DDL statements during the upgrading process, otherwise the undefined behavior error might occur.

3.9.1.2 Step 1: Install Ansible and dependencies on the Control Machine

> **Note:**
>
> If you have installed Ansible and its dependencies, you can skip this step.

TiDB Ansible release-3.0 depends on Ansible 2.4.2 ~ 2.7.11 (`2.4.2 ansible 2.7.11`, Ansible 2.7.11 recommended) and the Python modules of `jinja2 2.9.6` and `jmespath 0.9.0`.

To make it easy to manage dependencies, use `pip` to install Ansible and its dependencies. For details, see Install Ansible and its dependencies on the Control Machine. For offline environment, see Install Ansible and its dependencies offline on the Control Machine.

After the installation is finished, you can view the version information using the following command:

```
ansible --version
```

```
ansible 2.7.11
```

```
pip show jinja2
```

```
Name: Jinja2
Version: 2.10
```

```
pip show jmespath
```

```
Name: jmespath
Version: 0.9.0
```

Note:

- You must install Ansible and its dependencies following the above procedures.
- Make sure that the Jinja2 version is correct, otherwise an error occurs when you start Grafana.
- Make sure that the jmespath version is correct, otherwise an error occurs when you perform a rolling update to TiKV.

3.9.1.3 Step 2: Download TiDB Ansible to the Control Machine

1. Log in to the Control Machine using the `tidb` user account and enter the `/home/tidb` directory.
2. Back up the `tidb-ansible` folders of TiDB 2.0, 2.1, or an earlier 3.0 version using the following command:

   ```
   mv tidb-ansible tidb-ansible-bak
   ```

3. Download the tidb-ansible with the tag corresponding to TiDB 3.0. For more details, See Download TiDB Ansible to the Control Machine. The default folder name is `tidb-ansible`.

   ```
   git clone -b $tag https://github.com/pingcap/tidb-ansible.git
   ```

3.9.1.4 Step 3: Edit the `inventory.ini` file and the configuration file

Log in to the Control Machine using the `tidb` user account and enter the `/home/tidb/tidb-ansible` directory.

3.9.1.4.1 Edit the `inventory.ini` file

Edit the `inventory.ini` file. For IP information, see the `/home/tidb/tidb-ansible-bak/inventory.ini` backup file.

Note:

Pay special attention to the following variables configuration. For variable meaning, see Description of other variables.

1. Make sure that `ansible_user` is the normal user. For unified privilege management, remote installation using the root user is no longer supported. The default configuration uses the `tidb` user as the SSH remote user and the program running user.

```
## Connection
# ssh via normal user
ansible_user = tidb
```

   You can refer to How to configure SSH mutual trust and sudo rules on the Control Machine to automatically configure the mutual trust among hosts.

2. Keep the `process_supervision` variable consistent with that in the previous version. It is recommended to use `systemd` by default.

```
# process supervision, [systemd, supervise]
process_supervision = systemd
```

   If you need to modify this variable, see How to modify the supervision method of a process from `supervise` to `systemd`. Before you upgrade, first use the `/home/tidb/tidb-ansible-bak/` backup branch to modify the supervision method of a process.

3.9.1.4.2 Edit the configuration file of TiDB cluster components

If you have previously customized the configuration file of TiDB cluster components, refer to the backup file to modify the corresponding configuration file in the `/home/tidb/tidb-ansible/conf` directory.

Note the following parameter changes:

- In the TiKV configuration, `end-point-concurrency` is changed to three parameters: `high-concurrency`, `normal-concurrency` and `low-concurrency`.

```
readpool:
  coprocessor:
    # Notice: if CPU_NUM > 8, default thread pool size for coprocessors
    # will be set to CPU_NUM * 0.8.
    # high-concurrency: 8
    # normal-concurrency: 8
    # low-concurrency: 8
```

> **Note:**
> For the cluster topology of multiple TiKV instances (processes) on a single machine, you need to modify the three parameters above.

Recommended configuration: the number of TiKV instances * the parameter value = the number of CPU cores * 0.8.

- In the TiKV configuration, the `block-cache-size` parameter of different CFs is changed to `block-cache`.

```
storage:
  block-cache:
    capacity: "1GB"
```

> **Note:**
> For the cluster topology of multiple TiKV instances (processes) on a single machine, you need to modify the `capacity` parameter.

Recommended configuration: `capacity` = MEM_TOTAL * 0.5 / the number of TiKV instances.

- In the TiKV configuration, you need to configure the `tikv_status_port` port for the multiple instances on a single machine scenario. Before you configure it, check whether a port conflict exists.

```
[tikv_servers]
TiKV1-1 ansible_host=172.16.10.4 deploy_dir=/data1/deploy tikv_port
    ↪ =20171 tikv_status_port=20181 labels="host=tikv1"
TiKV1-2 ansible_host=172.16.10.4 deploy_dir=/data2/deploy tikv_port
    ↪ =20172 tikv_status_port=20182 labels="host=tikv1"
TiKV2-1 ansible_host=172.16.10.5 deploy_dir=/data1/deploy tikv_port
    ↪ =20171 tikv_status_port=20181 labels="host=tikv2"
TiKV2-2 ansible_host=172.16.10.5 deploy_dir=/data2/deploy tikv_port
    ↪ =20172 tikv_status_port=20182 labels="host=tikv2"
TiKV3-1 ansible_host=172.16.10.6 deploy_dir=/data1/deploy tikv_port
    ↪ =20171 tikv_status_port=20181 labels="host=tikv3"
TiKV3-2 ansible_host=172.16.10.6 deploy_dir=/data2/deploy tikv_port
    ↪ =20172 tikv_status_port=20182 labels="host=tikv3"
```

3.9.1.5 Step 4: Download TiDB 3.0 binary to the Control Machine

Make sure that `tidb_version = v3.0.x` in the `tidb-ansible/inventory.ini` file, and then run the following command to download TiDB 3.0 binary to the Control Machine:

```
ansible-playbook local_prepare.yml
```

3.9.1.6 Step 5: Perform a rolling update to TiDB cluster components

- If the `process_supervision` variable uses the default `systemd` parameter, perform a rolling update to the TiDB cluster using the following command corresponding to your current TiDB cluster version.
  - When the TiDB cluster version < 3.0.0, use `excessive_rolling_update.yml`.

    ```
    ansible-playbook excessive_rolling_update.yml
    ```

  - When the TiDB cluster version 3.0.0, use `rolling_update.yml` for both rolling updates and daily rolling restarts.

    ```
    ansible-playbook rolling_update.yml
    ```

- If the `process_supervision` variable uses the `supervise` parameter, perform a rolling update to the TiDB cluster using `rolling_update.yml`, no matter what version the current TiDB cluster is.

  ```
  ansible-playbook rolling_update.yml
  ```

3.9.1.7 Step 6: Perform a rolling update to TiDB monitoring components

```
ansible-playbook rolling_update_monitor.yml
```

3.10 Troubleshoot

3.10.1 TiDB Troubleshooting Map

This document summarizes common issues in TiDB and other components. You can use this map to diagnose and solve issues when you encounter related problems.

3.10.1.1 1. Service Unavailable

3.10.1.1.1 1.1 The client reports `Region is Unavailable` error

- 1.1.1 The `Region is Unavailable` error is usually because a Region is not available for a period of time. You might encounter `TiKV server is busy`, or the request to TiKV fails due to `not leader` or `epoch not match`, or the request to TiKV time out. In such cases, TiDB performs a `backoff` retry mechanism. When the `backoff` exceeds a threshold (20s by default), the error will be sent to the client. Within the `backoff` threshold, this error is not visible to the client.
- 1.1.2 Multiple TiKV instances are OOM at the same time, which causes no Leader in a Region for a period of time. See case-991 in Chinese.
- 1.1.3 TiKV reports `TiKV server is busy`, and exceeds the `backoff` time. For more details, refer to 4.3. `TiKV server is busy` is a result of the internal flow control mechanism and should not be counted in the `backoff` time. This issue will be fixed.
- 1.1.4 Multiple TiKV instances failed to start, which causes no Leader in a Region. When multiple TiKV instances are deployed in a physical machine, the failure of the physical machine can cause no Leader in a Region if the label is not properly configured. See case-228 in Chinese.
- 1.1.5 When a Follower apply is lagged in a previous epoch, after the Follower becomes a Leader, it rejects the request with `epoch not match`. See case-958 in Chinese (TiKV needs to optimize its mechanism).

3.10.1.1.2 1.2 PD errors cause service unavailable

Refer to 5 PD issues.

3.10.1.2 2. Latency increases significantly

3.10.1.2.1 2.1 Transient increase

- 2.1.1 Wrong TiDB execution plan causes latency increase. Refer to 3.3.
- 2.1.2 PD Leader election issue or OOM. Refer to 5.2 and 5.3.
- 2.1.3 A significant number of Leader drops in some TiKV instances. Refer to 4.4.

3.10.1.2.2 2.2 Persistent and significant increase

- 2.2.1 TiKV single thread bottleneck
  - Too many Regions in a TiKV instance causes a single gRPC thread to be the bottleneck (Check the **Grafana -> TiKV-details -> Thread CPU/gRPC CPU Per Thread** metric). In v3.x or later versions, you can enable `Hibernate Region` to resolve the issue. See case-612 in Chinese.

  - For versions earlier than v3.0, when the raftstore thread or the apply thread becomes the bottleneck (**Grafana -> TiKV-details -> Thread CPU/raft store CPU** and **Async apply CPU** metrics exceed `80%`), you can scale out TiKV (v2.x) instances or upgrade to v3.x with multi-threading.

- 2.2.2 CPU load increases.
- 2.2.3 TiKV slow write. Refer to 4.5.
- 2.2.4 TiDB wrong execution plan. Refer to 3.3.

3.10.1.3 3. TiDB issues

3.10.1.3.1 3.1 DDL

- 3.1.1 An error `ERROR 1105 (HY000): unsupported modify decimal column precision` is reported when you modify the length of the `decimal` field. TiDB does not support changing the length of the `decimal` field.
- 3.1.2 TiDB DDL job hangs or executes slowly (use `admin show ddl jobs` to check DDL progress)
  - Cause 1: Network issue with other components (PD/TiKV).
  - Cause 2: Early versions of TiDB (earlier than v3.0.8) have heavy internal load because of a lot of goroutine at high concurrency.
  - Cause 3: In early versions (v2.1.15 & versions < v3.0.0-rc1), PD instances fail to delete TiDB keys, which causes every DDL change to wait for two leases.
  - For other unknown causes, report a bug.
  - Solution:
    * For cause 1, check the network connection between TiDB and TiKV/PD.
    * For cause 2 and 3, the issues are already fixed in later versions. You can upgrade TiDB to a later version.
    * For other causes, you can use the following solution of migrating the DDL owner.
  - DDL owner migration:
    * If you can connect to the TiDB server, execute the owner election command again: `curl -X POST http://{TiDBIP}:10080/ddl/owner/resign`
    * If you cannot connect to the TiDB server, use `tidb-ctl` to delete the DDL owner from the etcd of the PD cluster to trigger re-election: `tidb-ctl etcd delowner [LeaseID] [flags] + ownerKey`
- 3.1.3 TiDB reports `information schema is changed` error in log

- Cause 1: The DML operation touches a table that is under DDL. You can use `admin show ddl job` to check the DDLs that are currently in progress.
- Cause 2: The current DML operation is executed too long. During the time, many DDL operations are executed, which causes `schema version` changes to be more than 1024. The new version `lock table` might also cause schema version changes.
- Cause 3: The TiDB instance that is currently executing DML statements cannot load the new `schema information` (maybe caused by network issues with PD or TiKV). During this time, many DDL statements are executed (including `lock table`), which causes `schema version` changes to be more than 1024.
- Solution: The first two causes do not impact the application, as the related DML operations retry after failure. For cause 3, you need to check the network between TiDB and TiKV/PD.
- Background: The increased number of `schema version` is consistent with the number of `schema state` of each DDL change operation. For example, the `create table` operation has 1 version change, and the `add column` operation has 4 version changes. Therefore, too many column change operations might cause `schema version` to increase fast. For details, refer to online schema change.

- 3.1.4 TiDB reports `information schema is out of date` in log

  - Cause 1: The TiDB server that is executing the DML statement is stopped by `graceful kill` and prepares to exit. The execution time of the transaction that contains the DML statement exceeds one DDL lease. An error is reported when the transaction is committed.
  - Cause 2: The TiDB server cannot connect to PD or TiKV when it is executing the DML statement, which causes the following problems:
    * The TiDB server did not load the new schema within one DDL lease (`45s` by default); or
    * The TiDB server disconnects from PD with the `keep alive` setting.
  - Cause 3: TiKV has high load or network timed out. Check the node loads in **Grafana** -> **TiDB** and **TiKV**.
  - Solution:
    * For cause 1, retry the DML operation when TiDB is started.
    * For cause 2, check the network between the TiDB server and PD/TiKV.
    * For cause 3, investigate why TiKV is busy. Refer to 4 TiKV issues.

3.10.1.3.2 3.2 OOM issues

- 3.2.1 Symptom

  - Client: The client reports the error `ERROR 2013 (HY000): Lost connection to MySQL server during query`.
  - Check the log
    * Execute `dmesg -T | grep tidb-server`. The result shows the OOM-killer log around the time point when the error occurs.
    * Grep the "Welcome to TiDB" log in `tidb.log` around the time point after the error occurs (namely, the time when tidb-server restarts).
    * Grep `fatal error: runtime: out of memory` or `cannot allocate memory` in `tidb_stderr.log`.
    * In v2.1.8 or earlier versions, you can grep `fatal error: stack overflow` in the `tidb_stderr.log`.
  - Monitor: The memory usage of tidb-server instances increases sharply in a short period of time.

- 3.2.2 Locate the SQL statement that causes OOM. (Currently all versions of TiDB cannot locate SQL accurately. You still need to analyze whether OOM is caused by the SQL statement after you locate one.)

  - For versions >= v3.0.0, grep "expensive_query" in `tidb.log`. That log message records SQL queries that timed out or exceed memory quota.
  - For versions < v3.0.0, grep "memory exceeds quota" in `tidb.log` to locate SQL queries that exceed memory quota.

> **Note:**
>
> The default threshold for a single SQL memory usage is `32GB` (in bytes, scope:`SESSION`). You can set this parameter by configuring `tidb_mem_quota_query`. You can also modify the `mem-quota-query` item (in bytes) in the configuration file by hot loading the configuration items.

- 3.2.3 Mitigate OOM issues

  - By enabling `SWAP`, you can mitigate the OOM issue caused by overuse of memory by large queries. When the memory is insufficient, this method can have impact on the performance of large queries due to the I/O overhead. The degree to which the performance is affected depends on the remaining memory space and the disk I/O speed.

- 3.2.4 Typical reasons for OOM

- The SQL query has `join`. If you view the SQL statement by using `explain`, you can find that the `join` operation selects the `HashJoin` algorithm and the `inner` table is large.
- The data volume of a single `UPDATE/DELETE` query is too large. See case-882 in Chinese.
- The SQL contains multiple sub-queries connected by `Union`. See case-1828 in Chinese.

3.10.1.3.3 3.3 Wrong execution plan

- 3.3.1 Symptom
  - SQL query execution time is much longer compared with that of previous executions, or the execution plan suddenly changes. If the execution plan is logged in the slow log, you can directly compare the execution plans.
  - SQL query execution time is much longer compared with that of other databases such as MySQL. Compare the execution plan with other databases to see the differences, such as `Join Order`.
  - In slow log, the number of SQL execution time `Scan Keys` is large.
- 3.3.2 Investigate the execution plan
  - `explain analyze {SQL}`. When the execution time is acceptable, compare `count` in the result of `explain analyze` and the number of `row` in `execution info`. If a large difference is found in the `TableScan/IndexScan` row, it is likely that the statistics is incorrect. If a large difference is found in other rows, the problem might not be in the statistics.
  - `select count(*)`. When the execution plan contains a `join` operation, `explain analyze` might take a long time. You can check whether the problem is in the statistics by executing `select count(*)` for the conditions on `TableScan/IndexScan` and comparing the `row count` information in the `explain` result.
- 3.3.3 Mitigation
  - For v3.0 and later versions, use the `SQL Bind` feature to bind the execution plan.
  - Update the statistics. If you are roughly sure that the problem is caused by the statistics, dump the statistics. If the cause is outdated statistics, such as the `modify count/row count` in `show stats_meta` is greater than a certain value (for example, 0.3), or the table has an index of time column, you can try recovering by using `analyze table`. If `auto analyze` is configured, check whether the `tidb_auto_analyze_ratio` system variable is too large (for example, greater than 0.3), and whether the current time is between `tidb_auto_analyze_start_time` and `tidb_auto_analyze_end_time`.
  - For other situations, report a bug.

3.10.1.3.4 3.4 SQL execution error

- 3.4.1 The client reports the `ERROR 1265(01000)Data Truncated` error. This is because the way TiDB internally calculates the precision of `Decimal` type is incompatible with that of MySQL. This issue has been fixed in v3.0.10 (#14438).

  - Cause:

    In MySQL, if two large-precision `Decimal` are divided and the result exceeds the maximum decimal precision (`30`), only `30` digits are reserved and no error is reported;

    In TiDB, the calculation result is the same as in MySQL, but inside the data structure that represents `Decimal`, a field for decimal precision still retains the actual precision.

    Take `(0.1^30)/ 10` as an example. The results in TiDB and MySQL are both `0`, because the precision is `30` at most. However, in TiDB, the field for decimal precision is still `31`.

    After multiple `Decimal` divisions, even though the result is correct, this precision field could grow larger and larger, and eventually exceeds the threshold in TiDB (`72`), and the `Data Truncated` error is reported.

    The multiplication of `Decimal` does not have this issue, because the out-of-bounds is bypassed, and the precision is set to the maximum precision limit.

  - Solution: You can bypass this issue by manually adding `Cast(xx as decimal(a , b))`, in which `a` and `b` are the target precisions.

3.10.1.4 4. TiKV issues

3.10.1.4.1 4.1 TiKV panics and fails to start

- 4.1.1 `sync-log = false`. The `unexpected raft log index: last_index X < applied_index Y` error is returned after the machine is powered off.

  This issue is expected. You can restore the Region using `tikv-ctl`.

- 4.1.2 If TiKV is deployed on a virtual machine, when the virtual machine is killed or the physical machine is powered off, the `entries[X, Y] is unavailable from storage` error is reported.

  This issue is expected. The `fsync` of virtual machines is not reliable, so you need to restore the Region using `tikv-ctl`.

- 4.1.3 For other unexpected causes, report a bug.

3.10.1.4.2 4.2 TiKV OOM

- 4.2.1 If the `block-cache` configuration is too large, it might cause OOM.

  To verify the cause of the problem, check the `block cache size` of RocksDB by selecting the corresponding instance in the monitor **Grafana** -> **TiKV-details**.

  Meanwhile, check whether the `[storage.block-cache] capacity = # "1GB"` ↪ parameter is set properly. By default, TiKV's `block-cache` is set to `45%` of the total memory of the machine. You need to explicitly specify this parameter when you deploy TiKV in the container, because TiKV obtains the memory of the physical machine, which might exceed the memory limit of the container.

- 4.2.2 Coprocessor receives many large queries and returns a large volume of data. gRPC fails to send data as quickly as the coprocessor returns data, which results in OOM.

  To verify the cause, you can check whether `response size` exceeds the `network` ↪ `outbound` traffic by viewing the monitor **Grafana** -> **TiKV-details** -> **coprocessor overview**.

- 4.2.3 Other components occupy too much memory.

  This issue is unexpected. You can report a bug.

3.10.1.4.3 4.3 The client reports the `server is busy` error

Check the specific cause for busy by viewing the monitor **Grafana** -> **TiKV** -> **errors**. `server is busy` is caused by the flow control mechanism of TiKV, which informs `tidb/ti` ↪ `-client` that TiKV is currently under too much pressure and will retry later.

- 4.3.1 TiKV RocksDB encounters `write stall`.

  A TiKV instance has two RocksDB instances, one in `data/raft` to save the Raft log, another in `data/db` to save the real data. You can check the specific cause for stall by running `grep "Stalling" RocksDB` in the log. The RocksDB log is a file starting with `LOG`, and `LOG` is the current log.

  - Too many `level0 sst` causes stall. You can add the `[rocksdb] max-sub-` ↪ `compactions = 2` (or 3) parameter to speed up `level0 sst` compaction. The compaction task from level0 to level1 is divided into several subtasks (the max number of subtasks is the value of `max-sub-compactions`) to be executed concurrently. See case-815 in Chinese.
  - Too many `pending compaction bytes` causes stall. The disk I/O fails to keep up with the write operations in business peaks. You can mitigate this problem by increasing the `soft-pending-compaction-bytes-limit` and `hard-pending-` ↪ `compaction-bytes-limit` of the corresponding CF.

    * The default value of `[rocksdb.defaultcf] soft-pending-compaction-bytes-limit` is `64GB`. If the pending compaction bytes reaches the threshold, RocksDB slows down the write speed. You can set `[rocksdb.defaultcf] soft-pending-compaction-bytes-limit` to `128GB`.
    * The default value of `hard-pending-compaction-bytes-limit` is `256GB`. If the pending compaction bytes reaches the threshold (this is not likely to happen, because RocksDB slows down the write after the pending compaction bytes reaches `soft-pending-compaction-bytes-limit`), RocksDB stops the write operation. You can set `hard-pending-compaction-bytes-limit` to `512GB`.
    * If the disk I/O capacity fails to keep up with the write for a long time, it is recommended to scale up your disk. If the disk throughput reaches the upper limit and causes write stall (for example, the SATA SSD is much lower than NVME SSD), while the CPU resources is sufficient, you may apply a compression algorithm of higher compression ratio. This way, the CPU resources is traded for disk resources, and the pressure on the disk is eased.
    * If the default CF compaction sees a high pressure, change the `[rocksdb.defaultcf] compression-per-level` parameter from `["no", "no", "lz4", "lz4", "lz4", "zstd", "zstd"]` to `["no", "no", "zstd", "zstd", "zstd", "zstd", "zstd"]`.
  - Too many memtables causes stall. This usually occurs when the amount of instant writes is large and the memtables flush to the disk slowly. If the disk write speed cannot be improved, and this issue only occurs during business peaks, you can mitigate it by increasing the `max-write-buffer-number` of the corresponding CF.
    * For example, set `[rocksdb.defaultcf] max-write-buffer-number` to `8` (`5` by default). Note that this might cause more memory usage in the peak, because more memtables might be in the memory.

- 4.3.2 `scheduler too busy`

  - Serious write conflict. `latch wait duration` is high. You can view `latch wait duration` in the monitor **Grafana -> TiKV-details -> scheduler prewrite/scheduler commit**. When the write tasks pile up in the scheduler, the pending write tasks exceed the threshold set in `[storage] scheduler-pending-write-threshold` (100MB). You can verify the cause by viewing the metric corresponding to `MVCC_CONFLICT_COUNTER`.
  - Slow write causes write tasks to pile up. The data being written to TiKV exceeds the threshold set by `[storage] scheduler-pending-write-threshold` (100MB). Refer to 4.5.

- 4.3.3 `raftstore is busy`. The processing of messages is slower than the receiving of messages. The short-term `channel full` status does not affect the service, but if the error persists for a long time, it might cause Leader switch.

  - `append log` encounters stall. Refer to 4.3.1.
  - `append log duration` is high, which causes slow processing of messages. You can refer to 4.5 to analyze why `append log duration` is high.
  - raftstore receives a large batch of messages in an instant (check in the TiKV Raft messages dashboard), and fails to process them. Usually the short-term `channel full` status does not affect the service.

- 4.3.4 TiKV coprocessor is in a queue. The number of piled up tasks exceeds `coprocessor threads * readpool.coprocessor.max-tasks-per-worker-[normal |low|high]`. Too many large queries leads to the tasks piling up in coprocessor. You need to check whether a execution plan change causes a large number of table scan operations. Refer to 3.3.

3.10.1.4.4 4.4 Some TiKV nodes drop Leader frequently

- 4.4.1 Re-election because TiKV is restarted

  - After TiKV panics, it is pulled up by systemd and runs normally. You can check whether panic has occurred by viewing the TiKV log. Because this issue is unexpected, report a bug if it happens.
  - TiKV is stopped or killed by a third party and then pulled up by systemd. Check the cause by viewing `dmesg` and the TiKV log.
  - TiKV is OOM, which causes restart. Refer to 4.2.
  - TiKV is hung because of dynamically adjusting `THP` (Transparent Hugepage). See case case-500 in Chinese.

- 4.4.2 TiKV RocksDB encounters write stall and thus results in re-election. You can check if the monitor **Grafana** -> **TiKV-details** -> **errors** shows `server is busy`. Refer to 4.3.1.

- 4.4.3 Re-election because of network isolation.

3.10.1.4.5 4.5 TiKV write is slow

- 4.5.1 Check whether the TiKV write is low by viewing the `prewrite/commit/raw-put` duration of TiKV gRPC (only for raw KV clusters). Generally, you can locate the slow phase according to the performance-map. Some common situations are listed as follows.

- 4.5.2 The scheduler CPU is busy (only for transaction kv).

  The `scheduler command duration` of prewrite/commit is longer than the sum of `scheduler latch wait duration` and `storage async write duration`. The scheduler worker has a high CPU demand, such as over 80% of `scheduler-worker-pool-size` * 100%, or the CPU resources of the entire machine are relatively limited. If

the write workload is large, check if `[storage] scheduler-worker-pool-size` is set too small.

For other situations, report a bug.

- 4.5.3 Append log is slow.

  The **Raft IO**/`append log duration` in TiKV Grafana is high, usually because the disk write operation is slow. You can verify the cause by checking the `WAL Sync Duration max` value of RocksDB - raft.

  For other situations, report a bug.

- 4.5.4 The raftstore thread is busy.

  The **Raft Propose**/`propose wait duration` is significantly larger than the append log duration in TiKV Grafana. Take the following methods:

  - Check whether the `[raftstore] store-pool-size` configuration value is too small. It is recommended to set the value between `1` and `5` and not too large.
  - Check whether the CPU resources on the machine are insufficient.

- 4.5.5 Apply is slow.

  The **Raft IO**/`apply log duration` in TiKV Grafana is high, which usually comes with a high **Raft Propose**/`apply wait duration`. The possible causes are as follows:

  - `[raftstore] apply-pool-size` is too small (it is recommended to set the value between `1` and `5` and not too large), and the **Thread CPU**/`apply CPU` is large.
  - The CPU resources on the machine are insufficient.
  - Region write hot spot. A single apply thread has high CPU usage. Currently, we cannot properly address the hot spot problem on a single Region, which is being improved. To view the CPU usage of each thread, modify the Grafana expression and add `by (instance, name)`.
  - RocksDB write is slow. **RocksDB kv**/`max write duration` is high. A single Raft log might contain multiple KVs. When writing into RocksDB, 128 KVs are written into RocksDB in a write batch. Therefore, an apply log might be associated with multiple writes in RocksDB.
  - For other situations, report a bug.

- 4.5.6 Raft commit log is slow.

  The **Raft IO**/`commit log duration` in TiKV Grafana is high (this metric is only supported in Grafana after v4.x). Every Region corresponds to an independent Raft group. Raft has a flow control mechanism, similar to the sliding window mechanism of TCP. You can control the size of the sliding window by configuring the `[raftstore] raft-max-inflight-msgs = 256` parameter. If there is a write hot spot and the `commit log duration` is high, you can adjust the parameter, such as increasing it to `1024`.

- 4.5.7 For other situations, refer to the write path on performance-map and analyze the cause.

3.10.1.5 5. PD issues

3.10.1.5.1 5.1 PD scheduling

- 5.1.1 Merge
  - Empty Regions across tables cannot be merged. You need to modify the `[ ↪ coprocessor] split-region-on-table` parameter in TiKV, which is set to `false` in v4.x by default. See case-896 in Chinese.
  - Region merge is slow. You can check whether the merged operator is generated by accessing the monitor dashboard in **Grafana** -> **PD** -> **operator**. To accelerate the merge, increase the value of `merge-schedule-limit`.
- 5.1.2 Add replicas or take replicas online/offline
  - The TiKV disk uses 80% of the capacity, and PD does not add replicas. In this situation, the number of miss peers increases, so TiKV needs to be scaled out. See case-801 in Chinese.
  - When a TiKV node is taken offline, some Region cannot be migrated to other nodes. This issue has been fixed in v3.0.4 (#5526). See case-870 in Chinese.
- 5.1.3 Balance
  - The Leader/Region count is not evenly distributed. See case-394 and case-759 in Chinese. The major cause is that the balance performs scheduling based on the size of Region/Leader, so this might result in the uneven distribution of the count. In TiDB 4.0, the `[leader-schedule-policy]` parameter is introduced, which enables you to set the scheduling policy of Leader to be `count`-based or `size`-based.

3.10.1.5.2 5.2 PD election

- 5.2.1 PD switches Leader.
  - Cause 1: Disk. The disk where the PD node is located has full I/O load. Investigate whether PD is deployed with other components with high I/O demand and the health of the disk. You can verify the cause by viewing the monitor metrics in **Grafana** -> **disk performance** -> **latency/load**. You can also use the FIO tool to run a check on the disk if necessary. See case-292 in Chinese.

  - Cause 2: Network. The PD log shows `lost the TCP streaming connection`. You need to check whether there is a problem with the network between PD nodes and verify the cause by viewing `round trip` in the monitor **Grafana -> PD -> etcd**. See case-177 in Chinese.
  - Cause 3: High system load. The log shows `server is likely overloaded`. See case-214 in Chinese.

- 5.2.2 PD cannot elect a Leader or the election is slow.

  - PD cannot elect a Leader: The PD log shows `lease is not expired`. This issue has been fixed in v3.0.x and v2.1.19. See case-875 in Chinese.
  - The election is slow: The Region loading duration is long. You can check this issue by running `grep "regions cost"` in the PD log. If the result is in seconds, such as `load 460927 regions cost 11.77099s`, it means the Region loading is slow. You can enable the `region storage` feature in v3.0 by setting `use-region -storage` to `true`, which significantly reduce the Region loading duration. See case-429 in Chinese.

- 5.2.3 PD timed out when TiDB executes SQL statements.

  - PD doesn't have a Leader or switches Leader. Refer to 5.2.1 and 5.2.2.
  - Network issue. Check whether the network from TiDB to PD Leader is running normally by accessing the monitor **Grafana -> blackbox_exporter -> ping latency**.
  - PD panics. Report a bug.
  - PD is OOM. Refer to 5.3.
  - If the issue has other causes, get goroutine by running `curl http://127.0.0.1:2379/debug/pprof/goroutine?debug=2` and report a bug.

- 5.2.4 Other issues

  - PD reports the `FATAL` error, and the log shows `range failed to find revision pair`. This issue has been fixed in v3.0.8 (#2040). For details, see case-947 in Chinese.
  - For other situations, report a bug.

3.10.1.5.3 5.3 PD OOM

- 5.3.1 When the `/api/v1/regions` interface is used, too many Regions might cause PD OOM. This issue has been fixed in v3.0.8 (#1986).

- 5.3.2 PD OOM during the rolling upgrade. The size of gRPC messages is not limited, and the monitor shows that TCP InSegs is relatively large. This issue has been fixed in v3.0.6 (#1952).

3.10.1.5.4 5.4 Grafana display

- 5.4.1 The monitor in **Grafana** -> **PD** -> **cluster** -> **role** displays follower. The Grafana expression issue has been fixed in v3.0.8 (#1065). For details, see case-1022.

3.10.1.6 6. Ecosystem tools

3.10.1.6.1 6.1 TiDB Binlog

- 6.1.1 TiDB Binlog is a tool that collects changes from TiDB and provides backup and replication to downstream TiDB or MySQL platforms. For details, see TiDB Binlog on GitHub.
- 6.1.2 The `Update Time` in Pump/Drainer Status is updated normally, and no anomaly shows in the log, but no data is written to the downstream.
  - Binlog is not enabled in the TiDB configuration. Modify the `[binlog]` configuration in TiDB.
- 6.1.3 `sarama` in Drainer reports the `EOF` error.
  - The Kafka client version in Drainer is inconsistent with the version of Kafka. You need to modify the `[syncer.to] kafka-version` configuration.
- 6.1.4 Drainer fails to write to Kafka and panics, and Kafka reports the `Message was too large` error.
  - The binlog data is too large, so the single message written to Kafka is too large. You need to modify the following configuration of Kafka:

    ```
    message.max.bytes=1073741824
    replica.fetch.max.bytes=1073741824
    fetch.message.max.bytes=1073741824
    ```

    For details, see case-789 in Chinese.
- 6.1.5 Inconsistent data in upstream and downstream
  - Some TiDB nodes do not enable binlog. For v3.0.6 or later versions, you can check the binlog status of all the nodes by accessing the http://127.0.0.1:10080/info/all interface. For versions earlier than v3.0.6, you can check the binlog status by viewing the configuration file.
  - Some TiDB nodes go into the `ignore binlog` status. For v3.0.6 or later versions, you can check the binlog status of all the nodes by accessing the http://127.0.0.1:10080/info/all interface. For versions earlier than v3.0.6, check the TiDB log to see whether it contains the `ignore binlog` keyword.

  - The value of the timestamp column is inconsistent in upstream and downstream.
    * This is caused by different time zones. You need to ensure that Drainer is in the same time zone as the upstream and downstream databases. Drainer obtains its time zone from `/etc/localtime` and does not support the `TZ` environment variable. See case-826 in Chinese.
    * In TiDB, the default value of timestamp is `null`, but the same default value in MySQL 5.7 (not including MySQL 8) is the current time. Therefore, when the timestamp in upstream TiDB is `null` and the downstream is MySQL 5.7, the data in the timestamp column is inconsistent. You need to run `set @@global.explicit_defaults_for_timestamp=on;` in the upstream before enabling binlog.
  - For other situations, report a bug.

- 6.1.6 Slow replication
  - The downstream is TiDB/MySQL, and the upstream performs frequent DDL operations. See case-1023 in Chinese.
  - The downstream is TiDB/MySQL, and the table to be replicated has no primary key and no unique index, which causes reduced performance in binlog. It is recommended to add the primary key or unique index.
  - If the downstream outputs to files, check whether the output disk or network disk is slow.
  - For other situations, report a bug.

- 6.1.7 Pump cannot write binlog and reports the `no space left on device` error.
  - The local disk space is insufficient for Pump to write binlog data normally. You need to clean up the disk space and then restart Pump.

- 6.1.8 Pump reports the `fail to notify all living drainer` error when it is started.
  - Cause: When Pump is started, it notifies all Drainer nodes that are in the `online` state. If it fails to notify Drainer, this error log is printed.
  - Solution: Use the binlogctl tool to check whether each Drainer node is normal or not. This is to ensure that all Drainer nodes in the `online` state are working normally. If the state of a Drainer node is not consistent with its actual working status, use the binlogctl tool to change its state and then restart Pump. See the case fail-to-notify-all-living-drainer.

- 6.1.9 Drainer reports the `gen update sqls failed: table xxx: row data is corruption []` error.

  - Trigger: The upstream performs DML operations on this table while performing `DROP COLUMN` DDL. This issue has been fixed in v3.0.6. See case-820 in Chinese.

- 6.1.10 Drainer replication is hung. The process remains active but the checkpoint is not updated.

  - This issues has been fixed in v3.0.4. See case-741 in Chinese.

- 6.1.11 Any component panics.

  - Report a bug.

3.10.1.6.2 6.2 Data Migration

- 6.2.1 TiDB Data Migration (DM) is a migration tool that supports data migration from MySQL/MariaDB into TiDB. For details, see DM on GitHub.

- 6.2.2 `Access denied for user 'root'@'172.31.43.27' (using password: YES)` shows when you run `query status` or check the log.

  - The database related passwords in all the DM configuration files should be encrypted by `dmctl`. If a database password is empty, it is unnecessary to encrypt the password. Cleartext passwords can be used since v1.0.6.
  - During DM operation, the user of the upstream and downstream databases must have the corresponding read and write privileges. Data Migration also prechecks the corresponding privileges automatically while starting the data replication task.
  - To deploy different versions of DM-worker/DM-master/dmctl in a DM cluster, see the case study on AskTUG in Chinese.

- 6.2.3 A replication task is interrupted with the `driver: bad connection` error returned.

  - The `driver: bad connection` error indicates that an anomaly has occurred in the connection between DM and the downstream TiDB database (such as network failure, TiDB restart and so on), and that the data of the current request has not yet been sent to TiDB.
    * For versions earlier than DM 1.0.0 GA, stop the task by running `stop-task` and then restart the task by running `start-task`.
    * For DM 1.0.0 GA or later versions, an automatic retry mechanism for this type of error is added. See #265.

- 6.2.4 A replication task is interrupted with the `invalid connection` error.

  - The `invalid connection` error indicates that an anomaly has occurred in the connection between DM and the downstream TiDB database (such as network failure, TiDB restart, TiKV busy and so on), and that a part of the data for the current request has been sent to TiDB. Because DM has the feature of concurrently replicating data to the downstream in replication tasks, several errors might occur when a task is interrupted. You can check these errors by running `query-status` or `query-error`.
    * If only the `invalid connection` error occurs during the incremental replication process, DM retries the task automatically.
    * If DM does not retry or fails to retry automatically because of version problems (automatic retry is introduced in v1.0.0-rc.1), use `stop-task` to stop the task and then use `start-task` to restart the task.

- 6.2.5 The relay unit reports the error `event from * in * diff from passed-in event *`, or a replication task is interrupted with an error that fails to get or parse binlog, such as `get binlog error ERROR 1236 (HY000)and binlog checksum mismatch, data may be corrupted returned`

  - During the process that DM pulls relay log or the incremental replication, this two errors might occur if the size of the upstream binlog file exceeds 4 GB.
  - Cause: When writing relay logs, DM needs to perform event verification based on binlog positions and the binlog file size, and store the replicated binlog positions as checkpoints. However, the official MySQL uses uint32 to store binlog positions, which means the binlog position for a binlog file over 4 GB overflows, and then the errors above occur.
  - Solution:
    * For relay processing units, manually recover replication.
    * For binlog replication processing units, manually recover replication.

- 6.2.6 The DM replication is interrupted, and the log returns `ERROR 1236 (HY000) The slave is connecting using CHANGE MASTER TO MASTER_AUTO_POSITION = 1, but the master has purged binary logs containing GTIDs that the slave requires.`

  - Check whether the master binlog is purged.
  - Check the position information recorded in `relay.meta`.
    * `relay.meta` has recorded the empty GTID information. DM-worker saves the GTID information in memory to `relay.meta` when it exits or in every 30s. When DM-worker does not obtain the upstream GTID information, it saves the empty GTID information to `relay.meta`. See case-772 in Chinese.
    * The binlog event recorded in `relay.meta` triggers the incomplete recover process and records the wrong GTID information. This issue is fixed in v1.0.2, and might occur in earlier versions.

- 6.2.7 The DM replication process returns an error `Error 1366: incorrect utf8 value eda0bdedb29d(\ufffd\ufffd\ufffd\ufffd\ufffd\ufffd)`.

  - This value cannot be successfully written into MySQL 8.0 or TiDB, but can be written into MySQL 5.7. You can skip the data format check by enabling the `tidb_skip_utf8_check` parameter.

3.10.1.6.3 6.3 TiDB Lightning

- 6.3.1 TiDB Lightning is a tool for fast full import of large amounts of data into a TiDB cluster. See TiDB Lightning on GitHub.
- 6.3.2 Import speed is too slow.

  - `region-concurrency` is set too high, which causes thread contention and reduces performance. Three ways to troubleshoot:
    * The setting can be found from the start of the log by searching `region-concurrency`.
    * If TiDB Lightning shares a server with other services (for example, Importer), you must manually set `region-concurrency` to 75% of the total number of CPU cores on that server.
    * If there is a quota on CPU (for example, limited by Kubernetes settings), TiDB Lightning might not be able to read this out. In this case, `region-concurrency` must also be manually reduced.
  - Every additional index introduces a new KV pair for each row. If there are N indices, the actual size to be imported would be approximately (N+1) times the size of the Mydumper output. If the indices are negligible, you may first remove them from the schema, and add them back via `CREATE INDEX` after the import is complete.
  - The version of TiDB Lightning is old. Try the latest version, which might improve the import speed.

- 6.3.3 `checksum failed: checksum mismatched remote vs local`.

  - Cause 1: The table might already have data. These old data can affect the final checksum.
  - Cause 2: If the checksum of the target database is 0, which means nothing is imported, it is possible that the cluster is too hot and fails to take in any data.
  - Cause 3: If the data source is generated by the machine and not backed up by Mydumper, ensure it respects the constrains of the table. For example:
    * `AUTO_INCREMENT` columns need to be positive, and do not contain the value "0".
    * UNIQUE and PRIMARY KEYs must not have duplicate entries.

  - Solution: See Troubleshooting Solution.

- 6.3.4 `Checkpoint for ⋯ has invalid status:(error code)`

  - Cause: Checkpoint is enabled, and Lightning/Importer has previously abnormally exited. To prevent accidental data corruption, Lightning will not start until the error is addressed. The error code is an integer less than 25, with possible values as `0, 3, 6, 9, 12, 14, 15, 17, 18, 20 and 21`. The integer indicates the step where the unexpected exit occurs in the import process. The larger the integer is, the later the exit occurs.
  - Solution: See Troubleshooting Solution.

- 6.3.5 `ResourceTemporarilyUnavailable("Too many open engines ⋯: 8")`

  - Cause: The number of concurrent engine files exceeds the limit specified by tikv-importer. This could be caused by misconfiguration. In addition, even when the configuration is correct, if tidb-lightning has exited abnormally before, an engine file might be left at a dangling open state, which could cause this error as well.
  - Solution: See Troubleshooting Solution.

- 6.3.6 `cannot guess encoding for input file, please convert to UTF-8 manually`

  - Cause: TiDB Lightning only supports the UTF-8 and GB-18030 encodings. This error means the file is not in any of these encodings. It is also possible that the file has mixed encoding, such as containing a string in UTF-8 and another string in GB-18030, due to historical ALTER TABLE executions.
  - Solution: See Troubleshooting Solution.

- 6.3.7 `[sql2kv] sql encode error = [types:1292]invalid time format: '{1970 1 1 0 45 0 0}'`

  - Cause: A timestamp type entry has a time value that does not exist. This is either because of DST changes or because the time value has exceeded the supported range (from Jan 1, 1970 to Jan 19, 2038).
  - Solution: See Troubleshooting Solution.

3.10.1.7 7. Common log analysis

3.10.1.7.1 7.1 TiDB

- 7.1.1 `GC life time is shorter than transaction duration`.

  The transaction duration exceeds the GC lifetime (10 minutes by default).

  You can increase the GC lifetime by modifying the `mysql.tidb` table. Generally, it is not recommended to modify this parameter, because changing it might cause many old versions to pile up if this transaction has a large number of `update` and `delete` statements.

- 7.1.2 `txn takes too much time`.

  This error is returned when you commit a transaction that has not been committed for a long time (over 590 seconds).

  If your application needs to execute a transaction of such a long time, you can increase the `[tikv-client] max-txn-time-use = 590` parameter and the GC lifetime to avoid this issue. It is recommended to check whether your application needs such a long transaction time.

- 7.1.3 `coprocessor.go` reports `request outdated`.

  This error is returned when the coprocessor request sent to TiKV waits in a queue at TiKV for over 60 seconds.

  You need to investigate why the TiKV coprocessor is in a long queue.

- 7.1.4 `region_cache.go` reports a large number of `switch region peer to next due to send request fail`, and the error message is `context deadline exceeded`.

  The request for TiKV timed out and triggers the region cache to switch the request to other nodes. You can continue to run the `grep "<addr> cancelled` command on the `addr` field in the log and take the following steps according to the `grep` results:

  - `send request is cancelled`: The request timed out during the sending phase. You can investigate the monitoring **Grafana** -> **TiDB** -> **Batch Client**/`Pending Request Count by TiKV` and see whether the Pending Request Count is greater than 128:
    * If the value is greater than 128, the sending goes beyond the processing capacity of KV, so the sending piles up.
    * If the value is not greater than 128, check the log to see if the report is caused by the operation and maintenance changes of the corresponding KV; otherwise, this error is unexpected, and you need to report a bug.
  - `wait response is cancelled`: The request timed out after it is sent to TiKV. You need to check the response time of the corresponding TiKV address and the Region logs in PD and KV at that time.

- 7.1.5 `distsql.go` reports `inconsistent index`.

  The data index seems to be inconsistent. Run the `admin check table <TableName>` command on the table where the reported index is. If the check fails, close GC by running the following command, and report a bug:

```
begin;
update mysql.tidb set variable_value='72h' where variable_name='
    ↪ tikv_gc_life_time';
commit;
```

3.10.1.7.2 7.2 TiKV

- 7.2.1 `key is locked`.

  The read and write have conflict. The read request encounters data that has not been committed and needs to wait until the data is committed.

  A small number of this error has no impact on the business, but a large number of this error indicates that the read-write conflict is severe in your business.

- 7.2.2 `write conflict`.

  This is the write-write conflict in optimistic transactions. If multiple transactions modify the same key, only one transaction succeed and other transactions automatically obtain the timestamp again and retry the operation, with no impact on the business.

  If the conflict is severe, it might cause transaction failure after multiple retries. In this case, it is recommended to use the pessimistic lock.

- 7.2.3 `TxnLockNotFound`.

  This transaction commit is too slow, which is rolled back by other transactions after TTL (3 seconds for a small transaction by default). This transaction will automatically retry, so the business is usually not affected.

- 7.2.4 `PessimisticLockNotFound`.

  Similar to `TxnLockNotFound`. The pessimistic transaction commit is too slow and thus rolled back by other transactions.

- 7.2.5 `stale_epoch`.

  The request epoch is outdated, so TiDB re-sends the request after updating the routing. The business is not affected. Epoch changes when Region has a split/merge operation or a replica is migrated.

- 7.2.6 `peer is not leader`.

  The request is sent to a replica that is not Leader. If the error response indicates which replica is the latest Leader, TiDB updates the local routing according the error and sends a new request to the latest Leader. Usually, the business is not affected.

In v3.0 and later versions, TiDB tries other peers if the request to the previous Leader fails, which might lead to frequent `peer is not leader` in TiKV log. You can check the `switch region peer to next due to send request fail` log of the corresponding Region in TiDB to determine the root cause of the sending failure. For details, refer to 7.1.4.

This error might also be returned if a Region has no Leader due to other reasons. For details, see 4.4.

3.10.2 TiDB Cluster Troubleshooting Guide

You can use this guide to help you diagnose and solve basic problems while using TiDB. If your problem is not resolved, please collect the following information and create an issue:

- The exact error message and the operations while the error occurs
- The state of all the components
- The `error/fatal/panic` information in the log of the component that reports the error
- The configuration and deployment topology
- The TiDB component related issue in `dmesg`

For other information, see Frequently Asked Questions (FAQ).

3.10.2.1 Cannot connect to the database

1. Make sure all the services are started, including `tidb-server`, `pd-server`, and `tikv-server`.

2. Use the `ps` command to check if all the processes are running.

    - If a certain process is not running, see the following corresponding sections to diagnose and solve the issue.
    - If all the processes are running, check the `tidb-server` log to see if the following messages are displayed:
        - InformationSchema is out of date: This message is displayed if the `tikv-server` cannot be connected. Check the state and log of `pd-server` and `tikv-server`.
        - panic: This message is displayed if there is an issue with the program. Please provide the detailed panic log and create an issue.

3. If the data is cleared and the services are re-deployed, make sure that:

    - All the data in `tikv-server` and `pd-server` are cleared. The specific data is stored in `tikv-server` and the metadata is stored in `pd-server`. If only one of the two servers is cleared, the data will be inconsistent.

- After the data in `pd-server` and `tikv-server` are cleared and the `pd-server` and `tikv-server` are restarted, the `tidb-server` must be restarted too. The cluster ID is randomly allocated when the `pd-server` is initialized. So when the cluster is re-deployed, the cluster ID changes and you need to restart the `tidb-server` to get the new cluster ID.

3.10.2.2 Cannot start `tidb-server`

See the following for the situations when the `tidb-server` cannot be started:

- Error in the startup parameters.

  See the TiDB configuration and options.

- The port is occupied.

  Use the `lsof -i:port` command to show all the networking related to a given port and make sure the port to start the `tidb-server` is not occupied.

- Cannot connect to `pd-server`.

  - Check if the network between TiDB and PD is running smoothly, including whether the network can be pinged or if there is any issue with the Firewall configuration.
  - If there is no issue with the network, check the state and log of the `pd-server` process.

3.10.2.3 Cannot start `tikv-server`

See the following for the situations when the `tikv-server` cannot be started:

- Error in the startup parameters: See the TiKV configuration and options.
- The port is occupied: Use the `lsof -i:port` command to show all the networking related to a given port and make sure the port to start the `tikv-server` is not occupied.
- Cannot connect to `pd-server`.

  - Check if the network between TiDB and PD is running smoothly, including whether the network can be pinged or if there is any issue with the Firewall configuration.
  - If there is no issue with the network, check the state and log of the `pd-server` process.

- The file is occupied. Do not open two TiKV files on one database file directory.

3.10.2.4 Cannot start `pd-server`

See the following for the situations when the `pd-server` cannot be started:

- Error in the startup parameters.

  See the PD configuration and options.

- The port is occupied.

  Use the `lsof -i:port` command to show all the networking related to a given port and make sure the port to start the `pd-server` is not occupied.

3.10.2.5 The TiDB/TiKV/PD process aborts unexpectedly

- Is the process started on the foreground? The process might exit because the client aborts.
- Is `nohup+&` run in the command line? This might cause the process to abort because it receives the hup signal. It is recommended to write and run the startup command in a script.

3.10.2.6 TiDB panic

Please provide the panic log and create an issue.

3.10.2.7 The connection is rejected

Make sure the network parameters of the operating system are correct, including but not limited to:

- The port in the connection string is consistent with the `tidb-server` starting port.
- The firewall is configured correctly.

3.10.2.8 Open too many files

Before starting the process, make sure the result of `ulimit -n` is large enough. It is recommended to set the value to `unlimited` or larger than `1000000`.

3.10.2.9 Database access times out and the system load is too high

First, check the slow query log and see if it is because of some inappropriate SQL statement. If you failed to solve the problem, provide the following information:

- The deployment topology
  - How many `tidb-server`/`pd-server`/`tikv-server` instances are deployed?

  - How are these instances distributed in the machines?

- The hardware configuration of the machines where these instances are deployed:
  - The number of CPU cores
  - The size of the memory
  - The type of the disk (SSD or Hard Drive Disk)
  - Are they physical machines or virtual machines?

- Are there other services besides the TiDB cluster?
- Are the `pd-server`s and `tikv-server`s deployed separately?
- What is the current operation?
- Check the CPU thread name using the `top -H` command.
- Are there any exceptions in the network or IO monitoring data recently?

4 Reference

4.1 SQL

4.1.1 MySQL Compatibility

TiDB supports both the MySQL wire protocol and the majority of its syntax. This means that you can use your existing MySQL connectors and clients, and your existing applications can often be migrated to TiDB without changing any application code.

Currently TiDB Server advertises itself as MySQL 5.7 and works with most MySQL database tools such as PHPMyAdmin, Navicat, MySQL Workbench, mysqldump, and Mydumper/myloader.

However, some features of MySQL are not supported. This could be because there is now a better way to solve the problem (such as XML functions superseded by JSON), or a lack of current demand versus effort required (such as stored procedures and functions). Some features might also be difficult to implement as a distributed system.

> **Note:**
>
> This page refers to general differences between MySQL and TiDB. Refer to the dedicated pages for Security and Pessimistic Transaction Model compatibility.

4.1.1.1 Unsupported features

- Stored procedures and functions
- Triggers

- Events
- User-defined functions
- `FOREIGN KEY` constraints #18209
- Temporary tables #1248
- `FULLTEXT`/`SPATIAL` functions and indexes #1793
- Character sets other than `utf8`, `utf8mb4`, `ascii`, `latin1` and `binary`
- Collations other than `BINARY`
- Add/drop primary key
- SYS schema
- Optimizer trace
- XML Functions
- X-Protocol #1109
- Savepoints #6840
- Column-level privileges #9766
- `XA` syntax (TiDB uses a two-phase commit internally, but this is not exposed via an SQL interface)
- `CREATE TABLE tblName AS SELECT stmt` syntax #4754
- `CHECK TABLE` syntax #4673
- `CHECKSUM TABLE` syntax #1895
- `GET_LOCK` and `RELEASE_LOCK` functions #14994

4.1.1.2 Features that are different from MySQL

4.1.1.2.1 Auto-increment ID

In TiDB, auto-increment columns are only guaranteed to be unique and incremental on a single TiDB server, but they are *not* guaranteed to be incremental among multiple TiDB servers or allocated sequentially. Currently, TiDB allocates IDs in batches. If you insert data on multiple TiDB servers at the same time, the allocated IDs are not continuous. You can use the `tidb_allow_remove_auto_inc` system variable to enable or disable deleting the `AUTO_INCREMENT` attribute of a column. The syntax for deleting this column attribute is `alter table modify` or `alter table change`.

> **Note:**
>
> If you use auto-increment IDs in a cluster with multiple tidb-server instances, do not mix default values and custom values. Otherwise, an error might occur in the following situation.

Assume that you have a table with the auto-increment ID:

```
CREATE TABLE t(id int unique key AUTO_INCREMENT, c int);
```

The principle of the auto-increment ID in TiDB is that each tidb-server instance caches a section of ID values (currently 30000 IDs are cached) for allocation and fetches the next section after this section is used up.

Assume that the cluster contains two tidb-server instances, namely Instance A and Instance B. Instance A caches the auto-increment ID of [1, 30000], while Instance B caches the auto-increment ID of [30001, 60000].

The operations are executed as follows:

1. The client issues the `INSERT INTO t VALUES (1, 1)` statement to Instance B which sets the `id` to 1 and the statement is executed successfully.
2. The client issues the `INSERT INTO t (c)(1)` statement to Instance A. This statement does not specify the value of `id`, so Instance A allocates the value. Currently, Instances A caches the auto-increment ID of [1, 30000], so it allocates the `id` value to 1 and adds 1 to the local counter. However, at this time the data with the `id` of 1 already exists in the cluster, therefore it reports `Duplicated Error`.

Also, starting with TiDB 3.0.4, TiDB supports using the system variable `tidb_allow_remove_auto_in`
↪ to control whether the `AUTO_INCREMENT` property of a column is allowed to be removed by executing `ALTER TABLE MODIFY` or `ALTER TABLE CHANGE` statements. It is not allowed by default. Once the `AUTO_INCREMENT` property is removed, it cannot be recovered, because TiDB does not support adding the `AUTO_INCREMENT` column attribute.

> **Note:**
>
> If the primary key is not specified, TiDB uses the `_tibd_rowid` column to identify rows. The values of the `_tibd_rowid` column and the auto-increment column (if there is) are assigned by the same allocator. If the auto-increment column is specified as the primary key, then TiDB uses this column to identify rows. Therefore, there might be the following situations.

```
mysql> create table t(id int unique key AUTO_INCREMENT);
Query OK, 0 rows affected (0.05 sec)

mysql> insert into t values(),(),();
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> select _tidb_rowid, id from t;
+-------------+------+
| _tidb_rowid | id |
+-------------+------+
```

```
|          4 |   1 |
|          5 |   2 |
|          6 |   3 |
+-------------+------+
3 rows in set (0.01 sec)
```

In early versions, the cache size of auto-increment IDs in TiDB is transparent to users. Since v3.0.14, v3.1.2, and v4.0.rc-2, the `AUTO_ID_CACHE` table option has been introduced to allow users to customize the cache size of auto-increment IDs to be allocated. This cache size might be consumed by both auto-increment columns and `_tidb_rowid`. In addition, if the length of continuous IDs needed for an `INSERT` statement exceeds the value set by `AUTO_ID_CACHE`, TiDB will properly increase the cache size so that this insertion can be executed successfully.

4.1.1.2.2 Performance schema

Performance schema tables return empty results in TiDB. TiDB uses a combination of Prometheus and Grafana for performance metrics instead.

TiDB supports the `events_statements_summary_by_digest` table from TiDB 3.0.4. For more information, see Statement Summary Table.

4.1.1.2.3 Query Execution Plan

The output format of Query Execution Plan (`EXPLAIN`/`EXPLAIN FOR`) in TiDB is greatly different from that in MySQL. Besides, the output content and the privileges setting of `EXPLAIN FOR` are not the same as those of MySQL. See Understand the Query Execution Plan for more details.

4.1.1.2.4 Built-in functions

TiDB supports most of the MySQL built-in functions, but not all. See TiDB SQL Grammar for the supported functions.

4.1.1.2.5 DDL

In TiDB DDL does not block reads or writes to tables while in operation. However, some restrictions currently apply to DDL changes:

- Add Index:
  - Does not support creating multiple indexes at the same time.
  - Does not support the `VISIBLE/INVISIBLE` index.
  - Adding an index on a generated column via `ALTER TABLE` is not supported.
  - Other Index Type (HASH/BTREE/RTREE) is supported in syntax, but not applicable.

- Add Column:
  - Does not support creating multiple columns at the same time.
  - Does not support setting a column as the `PRIMARY KEY`, or creating a unique index, or specifying `AUTO_INCREMENT` while adding it.
- Drop Column: Does not support dropping the `PRIMARY KEY` column or index column.
- Change/Modify Column:
  - Does not support lossy changes, such as from `BIGINT` to `INTEGER` or `VARCHAR(255)` ↪ to `VARCHAR(10)`. Otherwise, the `length %d is less than origin %d` error might be output.
  - Does not support modifying the precision of `DECIMAL` data types.
  - Changing the field type to its superset is unsupported. For example, TiDB does not support changing the field type from `INTEGER` to `VARCHAR`, or from `TIMESTAMP` to `DATETIME`. Otherwise, the error information `Unsupported modify column:` ↪ `type %d not match origin %d` might be output.
  - Does not support changing the `UNSIGNED` attribute.
  - Only supports changing the `CHARACTER SET` attribute from `utf8` to `utf8mb4`.
- `LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}`: the syntax is supported, but is not applicable to TiDB. All DDL changes that are supported do not lock the table.
- `ALGORITHM [=] {DEFAULT|INSTANT|INPLACE|COPY}`: the syntax for `ALGORITHM=` ↪ `INSTANT` and `ALGORITHM=INPLACE` is fully supported, but it works differently from MySQL because some operations that are `INPLACE` in MySQL are `INSTANT` in TiDB. The syntax `ALGORITHM=COPY` is not applicable to TIDB and returns a warning.
- Multiple operations cannot be completed in a single `ALTER TABLE` statement. For example, it's not possible to add multiple columns or indexes in a single statement.
- The following Table Options are not supported in syntax:
  - `WITH/WITHOUT VALIDATION`
  - `SECONDARY_LOAD/SECONDARY_UNLOAD`
  - `CHECK/DROP CHECK`
  - `STATS_AUTO_RECALC/STATS_SAMPLE_PAGES`
  - `SECONDARY_ENGINE`
  - `ENCRYPTION`
- The following Table Partition syntaxes are not supported:
  - `PARTITION BY LIST`
  - `PARTITION BY KEY`
  - `SUBPARTITION`

- {CHECK|EXCHANGE|OPTIMIZE|REPAIR|IMPORT|DISCARD|REBUILD|REORGANIZE|
  ↪ COALESCE} PARTITION

For more information, see Online Schema Changes.

4.1.1.2.6 Analyze table

ANALYZE TABLE works differently in TiDB than in MySQL, in that it is a relatively lightweight and short-lived operation in MySQL/InnoDB, while in TiDB it completely rebuilds the statistics for a table and can take much longer to complete.

4.1.1.2.7 Limitations of SELECT syntax

- The syntax SELECT ... INTO @variable is not supported.
- The syntax SELECT ... GROUP BY ... WITH ROLLUP is not supported.
- The syntax SELECT .. GROUP BY expr does not imply GROUP BY expr ORDER BY
  ↪ expr as it does in MySQL 5.7. TiDB instead matches the behavior of MySQL 8.0 and does not imply a default order.

4.1.1.2.8 Views

Views in TiDB are currently non-insertable and non-updatable.

4.1.1.2.9 Storage engines

For compatibility reasons, TiDB supports the syntax to create tables with alternative storage engines. Metadata commands describe tables as being of engine InnoDB:

```
CREATE TABLE t1 (a INT) ENGINE=MyISAM;
```

```
Query OK, 0 rows affected (0.14 sec)
```

```
SHOW CREATE TABLE t1;
```

```
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `a` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
1 row in set (0.00 sec)
```

Architecturally, TiDB does support a similar storage engine abstraction to MySQL, and user tables are created in the engine specified by the --store option used when you start tidb-server (typically tikv).

4.1.1.2.10 SQL modes

TiDB supports most SQL modes:

- The compatibility modes, such as `ORACLE` and `POSTGRESQL` are parsed but ignored. Compatibility modes are deprecated in MySQL 5.7 and removed in MySQL 8.0.
- The `ONLY_FULL_GROUP_BY` mode has minor semantic differences from MySQL 5.7.
- The `NO_DIR_IN_CREATE` and `NO_ENGINE_SUBSTITUTION` SQL modes in MySQL are accepted for compatibility, but are not applicable to TiDB.

4.1.1.2.11 Version-specific comments

TiDB executes all MySQL version-specific comments, regardless of the version they apply to. For example, the comment `/*!90000 */` would instruct a MySQL server less than 9.0 to not execute code. In TiDB this code will always be executed:

```
mysql 8.0.16> SELECT /*!90000 "I should not run", */ "I should run" FROM
    ↪ dual;
+--------------+
| I should run |
+--------------+
| I should run |
+--------------+
1 row in set (0.00 sec)

tidb> SELECT /*!90000 "I should not run", */ "I should run" FROM dual;
+------------------+--------------+
| I should not run | I should run |
+------------------+--------------+
| I should not run | I should run |
+------------------+--------------+
1 row in set (0.00 sec)
```

4.1.1.2.12 Default differences

- Default character set:
  - The default value in TiDB is `utf8mb4`.
  - The default value in MySQL 5.7 is `latin1`, but changes to `utf8mb4` in MySQL 8.0.
- Default collation:
  - The default collation of `utf8mb4` in TiDB is `utf8mb4_bin`.
  - The default collation of `utf8mb4` in MySQL 5.7 is `utf8mb4_general_ci`, but changes to `utf8mb4_0900_ai_ci` in MySQL 8.0.

  - You can use the `SHOW CHARACTER SET` statement to check the default collations of all character sets.

- Default value of `foreign_key_checks`:
  - The default value in TiDB is `OFF` and currently TiDB only supports `OFF`.
  - The default value in MySQL 5.7 is `ON`.

- Default SQL mode:
  - The default SQL mode in TiDB includes these modes: `ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION`.
  - The default SQL mode in MySQL:
    * The default SQL mode in MySQL 5.7 is the same as TiDB.
    * The default SQL mode in MySQL 8.0 includes these modes: `ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION`.

- Default value of `lower_case_table_names`:
  - The default value in TiDB is 2 and currently TiDB only supports 2.
  - The default value in MySQL:
    * On Linux: 0
    * On Windows: 1
    * On macOS: 2

- Default value of `explicit_defaults_for_timestamp`:
  - The default value in TiDB is `ON` and currently TiDB only supports `ON`.
  - The default value in MySQL:
    * For MySQL 5.7: `OFF`
    * For MySQL 8.0: `ON`

4.1.1.2.13 Date and Time

Named timezone

TiDB supports named timezones such as `America/Los_Angeles` without having to load the time zone information tables as in MySQL.

Because they are built-in, named time zones in TiDB might behave slightly differently to MySQL, and cannot be modified. For example, in TiDB the names are case-sensitive #8087.

Note:

TiKV calculates time-related expressions that can be pushed down to it. This calculation uses the built-in time zone rule and does not depend on the time zone rule installed in the system. If the time zone rule installed in the system does not match the version of the built-in time zone rule in TiKV, the time data that can be inserted might result in a statement error in a few cases.

For example, if the tzdata 2018a time zone rule is installed in the system, the time `1988-04-17 02:00:00` can be inserted into TiDB of the 3.0.0-rc.1 version when the time zone is set to Asia/Shanghai or the time zone is set to the local time zone and the local time zone is Asia/Shanghai. But reading this record might result in a statement error because this time does not exist in the Asia/Shanghai time zone according to the tzdata 2018i time zone rule used by TiKV 3.0.0-rc.1. Daylight saving time is one hour late.

The named timezone rules in TiKV of two versions are as follows:

- 3.0.0 RC.1 and later: tzdata 2018i
- 2.1.0 RC.1 and later: tzdata 2018e

Zero month and zero day

It is not recommended to unset the `NO_ZERO_DATE` and `NO_ZERO_IN_DATE` SQL modes, which are enabled by default in TiDB as in MySQL. While TiDB supports operating with these modes disabled, the TiKV coprocessor does not. Executing certain statements that push down date and time processing functions to TiKV might result in a statement error.

Handling of space at the end of string line

Currently, when inserting data, TiDB keeps the space at the end of the line for the `VARCHAR` type, and truncate the space for the `CHAR` type. In case there is no index, TiDB behaves exactly the same as MySQL.

If there is a `UNIQUE` index on the `VARCHAR` data, MySQL truncates the space at the end of the `VARCHAR` line before determining whether the data is duplicated, which is similar to the processing of the `CHAR` type, while TiDB keeps the space.

When making a comparison, MySQL first truncates the constant and the space at the end of the column, while TiDB keeps them to enable exact comparison.

4.1.1.2.14 Type system differences

The following column types are supported by MySQL, but not by TiDB:

- FLOAT4/FLOAT8
- FIXED (alias for DECIMAL)

- SERIAL (alias for BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE)
- SQL_TSI_* (including SQL_TSI_YEAR, SQL_TSI_MONTH, SQL_TSI_WEEK, SQL_TSI_DAY, SQL_TSI_HOUR, SQL_TSI_MINUTE and SQL_TSI_SECOND)

4.1.2 SQL Language Structure

4.1.2.1 Literal Values

TiDB literal values include character literals, numeric literals, time and date literals, hexadecimal, binary literals, and NULL literals. This document introduces each of these literal values.

This document describes String literals, Numeric literals, NULL values, Hexadecimal literals, Date and time literals, Boolean literals, and Bit-value literals.

4.1.2.1.1 String literals

A string is a sequence of bytes or characters, enclosed within either single quote `'` or double quote `"` characters. For example:

```
'example string'
"example string"
```

Quoted strings placed next to each other are concatenated to a single string. The following lines are equivalent:

```
'a string'
'a' ' ' 'string'
"a" ' ' "string"
```

If the `ANSI_QUOTES` SQL MODE is enabled, string literals can be quoted only within single quotation marks because a string quoted within double quotation marks is interpreted as an identifier.

The string is divided into the following two types:

- Binary string: It consists of a sequence of bytes, whose charset and collation are both `binary`, and uses **byte** as the unit when compared with each other.
- Non-binary string: It consists of a sequence of characters and has various charsets and collations other than `binary`. When compared with each other, non-binary strings use **characters** as the unit. A character might contain multiple bytes, depending on the charset.

A string literal may have an optional `character set introducer` and `COLLATE clause`, to designate it as a string that uses a specific character set and collation.

```
[_charset_name]'string' [COLLATE collation_name]
```

For example:

```
SELECT _latin1'string';
SELECT _binary'string';
SELECT _utf8'string' COLLATE utf8_bin;
```

You can use N'literal' (or n'literal') to create a string in the national character set. The following statements are equivalent:

```
SELECT N'some text';
SELECT n'some text';
SELECT _utf8'some text';
```

To represent some special characters in a string, you can use escape characters to escape:

Escape Characters	Meaning
\0	An ASCII NUL (X'00') character
\'	A single quote ' character
\"	A double quote " character
\b	A backspace character
\n	A line break (newline) character
\r	A carriage return character
\t	A tab character
\z	ASCII 26 (Ctrl + Z)
\\	A backslash \ character
\%	A % character
_	A _ character

If you want to represent " in the string surrounded by ', or ' in the string surrounded by ", you do not need to use escape characters.

For more information, see String Literals in MySQL.

4.1.2.1.2 Numeric literals

Numeric literals include integer and DECIMAL literals and floating-point literals.

Integer may include . as a decimal separator. Numbers may be preceded by - or + to indicate a negative or positive value respectively.

Exact-value numeric literals can be represented as `1, .2, 3.4, -5, -6.78, +9.10`.

Numeric literals can also be represented in scientific notation, such as `1.2E3, 1.2E-3, -1.2E3, -1.2E-3`.

For more information, see Numeric Literals in MySQL.

4.1.2.1.3 Date and time literals

Date and time literal values can be represented in several formats, such as quoted strings or as numbers. When TiDB expects a date, it interprets any of `'2017-08-24'`, `'20170824'` and `20170824` as a date.

TiDB supports the following date formats:

- `'YYYY-MM-DD'` or `'YY-MM-DD'`: The `-` delimiter here is not strict. It can be any punctuation. For example, `'2017-08-24'`, `'2017&08&24'`, `'2012@12^31'` are all valid date formats. The only special punctuation is '.', which is is treated as a decimal point to separate the integer and fractional parts. Date and time can be separated by `T` or a white space. For example, `2017-8-24 10:42:00` and `2017-8-24T10:42:00` represents the same date and time.
- `'YYYYMMDDHHMMSS'` or `'YYMMDDHHMMSS'`: For example, `'20170824104520'` and `'170824104520'` are regarded as `'2017-08-24 10:45:20'`. However, if you provide a value out of range, such as `'170824304520'`, it is not treated as a valid date. Note that incorrect formats such as `YYYYMMDD HHMMSS`, `YYYYMMDD HH:MM:DD`, or `YYYY-MM-DD HHMMSS` will fail to insert.
- `YYYYMMDDHHMMSS` or `YYMMDDHHMMSS`: Note that these formats have no single or double quotes, but a number. For example, `20170824104520` is interpreted as `'2017-08-24 10:45:20'`.

DATETIME or TIMESTAMP values can be followed by a fractional part, used to represent microseconds precision (6 digits). The fractional part should always be separated from the rest of the time by a decimal point `.`.

The year value containing only two digits is ambiguous. It is recommended to use the four-digit year format. TiDB interprets the two-digit year value according to the following rules:

- If the year value is in the range of `70-99`, it is converted to `1970-1999`.
- If the year value is in the range of `00-69`, it is converted to `2000-2069`.

For month or day values less than 10, `'2017-8-4'` is the same as `'2017-08-04'`. The same is true for Time. For example, `'2017-08-24 1:2:3'` is the same as `'2017-08-24 01:02:03'`.

When the date or time value is required, TiDB selects the specified format according to the length of the value:

- 6 digits: `YYMMDD`.
- 12 digits: `YYMMDDHHMMSS`.
- 8 digits: `YYYYMMDD`.
- 14 digits: `YYYYMMDDHHMMSS`.

TiDB supports the following formats for time values:

- `'D HH:MM:SS'`, or `'HH:MM:SS'`, `'HH:MM'`, `'D HH:MM'`, `'D HH'`, `'SS'`: `D` means days and the valid value range is `0-34`.
- A number in `HHMMSS` format: For example, `231010` is interpreted as `'23:10:10'`.
- A number in any of `SS`, `MMSS`, and `HHMMSS`formats can be regarded as time.

The decimal point of the Time type is also `.`, with a precision of up to 6 digits after the decimal point.

See MySQL date and time literals for more details.

4.1.2.1.4 Boolean Literals

The constants `TRUE` and `FALSE` are equal to 1 and 0 respectively, which are not case sensitive.

```
SELECT TRUE, true, tRuE, FALSE, FaLsE, false;
```

```
+------+------+------+-------+-------+-------+
| TRUE | true | tRuE | FALSE | FaLsE | false |
+------+------+------+-------+-------+-------+
|    1 |    1 |    1 |     0 |     0 |     0 |
+------+------+------+-------+-------+-------+
1 row in set (0.00 sec)
```

4.1.2.1.5 Hexadecimal literals

Hexadecimal literal values are written using `X'val'` or `0xval` notation, where `val` contains hexadecimal digits. A leading `0x` is case sensitive and cannot be written as `0X`.

Legal hexadecimal literals:

```
X'ac12'
X'12AC'
x'ac12'
x'12AC'
0xac12
0x12AC
```

Illegal hexadecimal literals:

```
X'1z' (z is not a hexadecimal legal digit)
0X12AC (0X must be written as 0x)
```

Hexadecimal literals written using `X'val'` notation must contain an even number of digits. If the length of `val` is an odd number (for example, `X'A'` or `X'11A'`), to avoid the syntax error, pad the value with a leading zero:

```
mysql> select X'aff';
ERROR 1105 (HY000): line 0 column 13 near ""hex literal: invalid
    ↪ hexadecimal format, must even numbers, but 3 (total length 13)
mysql> select X'0aff';
+---------+
| X'0aff' |
+---------+
| 0x0aff |
+---------+
1 row in set (0.00 sec)
```

By default, a hexadecimal literal is a binary string.

To convert a string or a number to a string in hexadecimal format, use the `HEX()` function:

```
mysql> SELECT HEX('TiDB');
+-------------+
| HEX('TiDB') |
+-------------+
| 54694442  |
+-------------+
1 row in set (0.01 sec)

mysql> SELECT X'54694442';
+-------------+
| X'54694442' |
+-------------+
| TiDB      |
+-------------+
1 row in set (0.00 sec)
```

4.1.2.1.6 Bit-value literals

Bit-value literals are written using `b'val'` or `0bval` notation. The `val` is a binary value written using zeros and ones. A leading `0b` is case sensitive and cannot be written as `0B`.

Legal bit-value literals:

```
b'01'
B'01'
0b01
```

Illegal bit-value literals:

```
b'2' (2 is not a binary digit; it must be 0 or 1)
0B01 (0B must be written as 0b)
```

By default, a bit-value literal is a binary string.

Bit values are returned as binary values, which may not display well in the MySQL client. To convert a bit value to printable form, you can use a conversion function such as `BIN()` or `HEX()`.

```
CREATE TABLE t (b BIT(8));
INSERT INTO t SET b = b'00010011';
INSERT INTO t SET b = b'1110';
INSERT INTO t SET b = b'100101';

mysql> SELECT b+0, BIN(b), HEX(b) FROM t;
+------+--------+--------+
| b+0  | BIN(b) | HEX(b) |
+------+--------+--------+
|   19 | 10011  | 13     |
|   14 | 1110   | E      |
|   37 | 100101 | 25     |
+------+--------+--------+
3 rows in set (0.00 sec)
```

4.1.2.1.7 NULL Values

`NULL` means the data is empty, which is case-insensitive, and is synonymous with `\N` (case-sensitive).

> **Note:**
>
> `NULL` is not the same as `0`, nor the empty string `''`.

4.1.2.2 Schema Object Names

This document introduces schema object names in TiDB SQL statements.

Schema object names are used to name all schema objects in TiDB, including database, table, index, column, alias, and so on. You can quote these objects using identifiers in SQL statements.

You can use backticks to enclose the identifier. For example, `SELECT * FROM t` can also be written as `SELECT * FROM `t``. But if the identifier includes one or more special characters or is a reserved keyword, it must be enclosed in backticks to quote the schema object it represents.

```
SELECT * FROM `table` WHERE `table`.id = 20;
```

If you set `ANSI_QUOTES` in SQL MODE, TiDB will recognize the string enclosed in double quotation marks `"` as an identifier.

```
CREATE TABLE "test" (a varchar(10));
```

```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
    ↪ that corresponds to your TiDB version for the right syntax to use
    ↪ line 1 column 19 near ""test" (a varchar(10))"
```

```
SET SESSION sql_mode='ANSI_QUOTES';
```

```
Query OK, 0 rows affected (0.000 sec)
```

```
CREATE TABLE "test" (a varchar(10));
```

```
Query OK, 0 rows affected (0.012 sec)
```

If you want to use the backtick character in the quoted identifier, repeat the backtick twice. For example, to create a table a‘b:

```
CREATE TABLE `a``b` (a int);
```

In a `SELECT` statement, you can use an identifier or a string to specify an alias:

```
SELECT 1 AS `identifier`, 2 AS 'string';
```

```
+------------+--------+
| identifier | string |
+------------+--------+
|          1 |      2 |
+------------+--------+
1 row in set (0.00 sec)
```

For more information, see MySQL Schema Object Names.

4.1.2.2.1 Identifier qualifiers

Object names can be unqualified or qualified. For example, the following statement creates a table without a qualified name:

```
CREATE TABLE t (i int);
```

If you have not used the `USE` statement or the connection parameter to configure the database, the `ERROR 1046 (3D000): No database selected` error is displayed. At this time, you can specify the database qualified name:

```
CREATE TABLE test.t (i int);
```

White spaces can exist around `.` `table_name.col_name` and `table_name . col_name` are equivalent.

To quote this identifier, use:

```
`table_name`.`col_name`
```

Instead of:

```
`table_name.col_name`
```

For more information, see MySQL Identifier Qualifiers.

4.1.2.3 Keywords and Reserved Words

Keywords are words that have significance in SQL. Certain keywords, such as `SELECT`, `UPDATE`, or `DELETE`, are reserved and require special treatment for use as identifiers such as table and column names. For example, as table names, the reserved words must be quoted with backquotes:

```
mysql> CREATE TABLE select (a INT);
ERROR 1105 (HY000): line 0 column 19 near " (a INT)" (total length 27)
mysql> CREATE TABLE `select` (a INT);
Query OK, 0 rows affected (0.09 sec)
```

The `BEGIN` and `END` are keywords but not reserved words, so you do not need to quote them with backquotes:

```
mysql> CREATE TABLE `select` (BEGIN int, END int);
Query OK, 0 rows affected (0.09 sec)
```

Exception: A word that follows a period `.` qualifier does not need to be quoted with backquotes either:

```
mysql> CREATE TABLE test.select (BEGIN int, END int);
Query OK, 0 rows affected (0.08 sec)
```

The following list shows the keywords and reserved words in TiDB. Most of the reserved words are labelled with `(R)`. Window functions' reserved words are labelled with `(R-Window ↪ )`.

A

- ACTION
- ADD (R)
- ADDDATE
- ADMIN
- AFTER

- ALL (R)
- ALTER (R)
- ALWAYS
- ANALYZE(R)
- AND (R)
- ANY
- AS (R)
- ASC (R)
- ASCII
- AUTO_INCREMENT
- AVG
- AVG_ROW_LENGTH

B

- BEGIN
- BETWEEN (R)
- BIGINT (R)
- BINARY (R)
- BINLOG
- BIT
- BIT_XOR
- BLOB (R)
- BOOL
- BOOLEAN
- BOTH (R)
- BTREE
- BY (R)
- BYTE

C

- CASCADE (R)
- CASE (R)
- CAST
- CHANGE (R)
- CHAR (R)
- CHARACTER (R)
- CHARSET
- CHECK (R)
- CHECKSUM
- COALESCE
- COLLATE (R)
- COLLATION

- COLUMN (R)
- COLUMNS
- COMMENT
- COMMIT
- COMMITTED
- COMPACT
- COMPRESSED
- COMPRESSION
- CONNECTION
- CONSISTENT
- CONSTRAINT (R)
- CONVERT (R)
- COUNT
- CREATE (R)
- CROSS (R)
- CUME_DIST (R-Window)
- CURRENT_DATE (R)
- CURRENT_TIME (R)
- CURRENT_TIMESTAMP (R)
- CURRENT_USER (R)
- CURTIME

D

- DATA
- DATABASE (R)
- DATABASES (R)
- DATE
- DATE_ADD
- DATE_SUB
- DATETIME
- DAY
- DAY_HOUR (R)
- DAY_MICROSECOND (R)
- DAY_MINUTE (R)
- DAY_SECOND (R)
- DDL
- DEALLOCATE
- DEC
- DECIMAL (R)
- DEFAULT (R)
- DELAY_KEY_WRITE
- DELAYED (R)
- DELETE (R)

- DENSE_RANK (R-Window)
- DESC (R)
- DESCRIBE (R)
- DISABLE
- DISTINCT (R)
- DISTINCTROW (R)
- DIV (R)
- DO
- DOUBLE (R)
- DROP (R)
- DUAL (R)
- DUPLICATE
- DYNAMIC

E

- ELSE (R)
- ENABLE
- ENCLOSED
- END
- ENGINE
- ENGINES
- ENUM
- ESCAPE
- ESCAPED
- EVENTS
- EXCLUSIVE
- EXECUTE
- EXISTS
- EXPLAIN (R)
- EXTRACT

F

- FALSE (R)
- FIELDS
- FIRST
- FIRST_VALUE (R-Window)
- FIXED
- FLOAT (R)
- FLUSH
- FOR (R)
- FORCE (R)
- FOREIGN (R)

- FORMAT
- FROM (R)
- FULL
- FULLTEXT (R)
- FUNCTION

G

- GENERATED (R)
- GET_FORMAT
- GLOBAL
- GRANT (R)
- GRANTS
- GROUP (R)
- GROUP_CONCAT
- GROUPS (R-Window)

H

- HASH
- HAVING (R)
- HIGH_PRIORITY (R)
- HOUR
- HOUR_MICROSECOND (R)
- HOUR_MINUTE (R)
- HOUR_SECOND (R)

I

- IDENTIFIED
- IF (R)
- IGNORE (R)
- IN (R)
- INDEX (R)
- INDEXES
- INFILE (R)
- INNER (R)
- INSERT (R)
- INT (R)
- INTEGER (R)
- INTERVAL (R)
- INTO (R)
- IS (R)

- ISOLATION

J

- JOBS
- JOIN (R)
- JSON

K

- KEY (R)
- KEY_BLOCK_SIZE
- KEYS (R)
- KILL (R)

L

- LAG (R-Window)
- LAST_VALUE (R-Window)
- LEAD (R-Window)
- LEADING (R)
- LEFT (R)
- LESS
- LEVEL
- LIKE (R)
- LIMIT (R)
- LINES (R)
- LOAD (R)
- LOCAL
- LOCALTIME (R)
- LOCALTIMESTAMP (R)
- LOCK (R)
- LONGBLOB (R)
- LONGTEXT (R)
- LOW_PRIORITY (R)

M

- MAX
- MAX_ROWS
- MAXVALUE (R)
- MEDIUMBLOB (R)
- MEDIUMINT (R)

- MEDIUMTEXT (R)
- MICROSECOND
- MIN
- MIN_ROWS
- MINUTE
- MINUTE_MICROSECOND (R)
- MINUTE_SECOND (R)
- MIN
- MIN_ROWS
- MINUTE
- MINUTE_MICROSECOND
- MINUTE_SECOND
- MOD (R)
- MODE
- MODIRY
- MONTH

N

- NAMES
- NATIONAL
- NATURAL (R)
- NO
- NO_WRITE_TO_BINLOG (R)
- NONE
- NOT (R)
- NOW
- NTH_VALUE (R-Window)
- NTILE (R-Window)
- NULL (R)
- NUMERIC (R)
- NVARCHAR (R)

O

- OFFSET
- ON (R)
- ONLY
- OPTION (R)
- OR (R)
- ORDER (R)
- OUTER (R)
- OVER (R-Window)

P

- PARTITION (R)
- PARTITIONS
- PASSWORD
- PERCENT_RANK (R-Window)
- PLUGINS
- POSITION
- PRECISION (R)
- PREPARE
- PRIMARY (R)
- PRIVILEGES
- PROCEDURE (R)
- PROCESS
- PROCESSLIST

Q

- QUARTER
- QUERY
- QUICK

R

- RANGE (R)
- RANK (R-Window)
- READ (R)
- REAL (R)
- REDUNDANT
- REFERENCES (R)
- REGEXP (R)
- RENAME (R)
- REPEAT (R)
- REPEATABLE
- REPLACE (R)
- RESTRICT (R)
- REVERSE
- REVOKE (R)
- RIGHT (R)
- RLIKE (R)
- ROLLBACK
- ROW
- ROW_COUNT
- ROW_FORMAT

- ROW_NUMBER (R-Window)
- ROWS (R-Window)

S

- SCHEMA
- SCHEMAS
- SECOND
- SECOND_MICROSECOND (R)
- SELECT (R)
- SERIALIZABLE
- SESSION
- SET (R)
- SHARE
- SHARED
- SHOW (R)
- SIGNED
- SMALLINT (R)
- SNAPSHOT
- SOME
- SQL_CACHE
- SQL_CALC_FOUND_ROWS (R)
- SQL_NO_CACHE
- START
- STARTING (R)
- STATS
- STATS_BUCKETS
- STATS_HISTOGRAMS
- STATS_META
- STATS_PERSISTENT
- STATUS
- STORED (R)
- SUBDATE
- SUBSTR
- SUBSTRING
- SUM
- SUPER

T

- TABLE (R)
- TABLES
- TERMINATED (R)
- TEXT

- THAN
- THEN (R)
- TIDB
- TIDB_INLJ
- TIDB_SMJ
- TIME
- TIMESTAMP
- TIMESTAMPADD
- TIMESTAMPDIFF
- TINYBLOB (R)
- TINYINT (R)
- TINYTEXT (R)
- TO (R)
- TRAILING (R)
- TRANSACTION
- TRIGGER (R)
- TRIGGERS
- TRIM
- TRUE (R)
- TRUNCATE

U

- UNCOMMITTED
- UNION (R)
- UNIQUE (R)
- UNKNOWN
- UNLOCK (R)
- UNSIGNED (R)
- UPDATE (R)
- USE (R)
- USER
- USING (R)
- UTC_DATE (R)
- UTC_TIME (R)
- UTC_TIMESTAMP (R)

V

- VALUE
- VALUES (R)
- VARBINARY (R)
- VARCHAR (R)
- VARIABLES

- VIEW
- VIRTUAL (R)

W

- WARNINGS
- WEEK
- WHEN (R)
- WHERE (R)
- WINDOW (R-Window)
- WITH (R)
- WRITE (R)

X

- XOR (R)

Y

- YEAR
- YEAR_MONTH (R)

Z

- ZEROFILL (R)

4.1.2.4 User-Defined Variables

This document describes the concept of user-defined variables in TiDB and the methods to set and read the user-defined variables.

> **Warning:**
>
> User-defined variables are still an experimental feature. It is **NOT** recommended that you use them in the production environment.

The format of the user-defined variables is `@var_name`. The characters that compose `var_name` can be any characters that can compose an identifier, including the numbers `0-9` ↪ , the letters `a-zA-Z`, the underscore `_`, the dollar sign `$`, and the UTF-8 characters. In addition, it also includes the English period `.`. The user-defined variables are case-insensitive.

The user-defined variables are session-specific, which means a user variable defined by one client connection cannot be seen or used by other client connections.

4.1.2.4.1 Set the user-defined variables

You can use the `SET` statement to set a user-defined variable, and the syntax is `SET @var_name = expr [, @var_name = expr] ...;`. For example:

```
SET @favorite_db = 'TiDB';
```

```
SET @a = 'a', @b = 'b', @c = 'c';
```

For the assignment operator, you can also use `:=`. For example:

```
SET @favorite_db := 'TiDB';
```

The content to the right of the assignment operator can be any valid expression. For example:

```
SET @c = @a + @b;
```

```
set @c = b'1000001' + b'1000001';
```

4.1.2.4.2 Read the user-defined variables

To read a user-defined variable, you can use the `SELECT` statement to query:

```
SELECT @a1, @a2, @a3
```

```
+------+------+------+
| @a1 | @a2 | @a3 |
+------+------+------+
|   1 |   2 |   4 |
+------+------+------+
```

You can also assign values in the `SELECT` statement:

```
SELECT @a1, @a2, @a3, @a4 := @a1+@a2+@a3;
```

```
+------+------+------+--------------------+
| @a1 | @a2 | @a3 | @a4 := @a1+@a2+@a3 |
+------+------+------+--------------------+
|   1 |   2 |   4 |                  7 |
+------+------+------+--------------------+
```

Before the variable `@a4` is modified or the connection is closed, its value is always `7`。

If a hexadecimal literal or binary literal is used when setting the user-defined variable, TiDB will treat it as a binary string. If you want to set it to a number, you can manually add the `CAST` conversion, or use the numeric operator in the expression:

```
SET @v1 = b'1000001';
SET @v2 = b'1000001'+0;
SET @v3 = CAST(b'1000001' AS UNSIGNED);
```

```
SELECT @v1, @v2, @v3;
```

```
+------+------+------+
| @v1 | @v2 | @v3 |
+------+------+------+
| A    | 65   | 65   |
+------+------+------+
```

If you refer to a user-defined variable that has not been initialized, it has a value of NULL and a type of string.

```
SELECT @not_exist;
```

```
+------------+
| @not_exist |
+------------+
| NULL       |
+------------+
```

In addition to using the `SELECT` statement to read the user-defined variables, another common usage is the `PREPARE` statement. For example:

```
SET @s = 'SELECT SQRT(POW(?,2) + POW(?,2)) AS hypotenuse';
PREPARE stmt FROM @s;
SET @a = 6;
SET @b = 8;
EXECUTE stmt USING @a, @b;
```

```
+------------+
| hypotenuse |
+------------+
|         10 |
+------------+
```

The contents of the user-defined variables are not recognized as identifiers in the SQL statements. For example:

```
SELECT * from t;
```

```
+---+
| a |
```

```
+---+
| 1 |
+---+
```

```
SET @col = "`a`";
SELECT @col FROM t;
```

```
+------+
| @col |
+------+
| `a` |
+------+
```

For more information, see User-Defined Variables in MySQL.

4.1.2.5 Expression Syntax

The following rules define the expression syntax in TiDB. You can find the definition in `parser/parser.y`. The syntax parsing in TiDB is based on Yacc.

```
Expression:
     singleAtIdentifier assignmentEq Expression
    | Expression logOr Expression
    | Expression "XOR" Expression
    | Expression logAnd Expression
    | "NOT" Expression
    | Factor IsOrNotOp trueKwd
    | Factor IsOrNotOp falseKwd
    | Factor IsOrNotOp "UNKNOWN"
    | Factor

Factor:
     Factor IsOrNotOp "NULL"
    | Factor CompareOp PredicateExpr
    | Factor CompareOp singleAtIdentifier assignmentEq PredicateExpr
    | Factor CompareOp AnyOrAll SubSelect
    | PredicateExpr

PredicateExpr:
     PrimaryFactor InOrNotOp '(' ExpressionList ')'
    | PrimaryFactor InOrNotOp SubSelect
    | PrimaryFactor BetweenOrNotOp PrimaryFactor "AND" PredicateExpr
    | PrimaryFactor LikeOrNotOp PrimaryExpression LikeEscapeOpt
    | PrimaryFactor RegexpOrNotOp PrimaryExpression
    | PrimaryFactor
```

```
PrimaryFactor:
     PrimaryFactor '|' PrimaryFactor
    | PrimaryFactor '&' PrimaryFactor
    | PrimaryFactor "<<" PrimaryFactor
    | PrimaryFactor ">>" PrimaryFactor
    | PrimaryFactor '+' PrimaryFactor
    | PrimaryFactor '-' PrimaryFactor
    | PrimaryFactor '*' PrimaryFactor
    | PrimaryFactor '/' PrimaryFactor
    | PrimaryFactor '%' PrimaryFactor
    | PrimaryFactor "DIV" PrimaryFactor
    | PrimaryFactor "MOD" PrimaryFactor
    | PrimaryFactor '^' PrimaryFactor
    | PrimaryExpression

PrimaryExpression:
     Operand
    | FunctionCallKeyword
    | FunctionCallNonKeyword
    | FunctionCallAgg
    | FunctionCallGeneric
    | Identifier jss stringLit
    | Identifier juss stringLit
    | SubSelect
    | '!' PrimaryExpression
    | '~' PrimaryExpression
    | '-' PrimaryExpression
    | '+' PrimaryExpression
    | "BINARY" PrimaryExpression
    | PrimaryExpression "COLLATE" StringName
```

4.1.2.6 Comment Syntax

This document describes the comment syntax supported by TiDB.

TiDB supports three comment styles:

- Use # to comment a line:

```
SELECT 1+1;   # comments
```

```
+------+
| 1+1  |
+------+
```

```
|   2 |
+------+
1 row in set (0.00 sec)
```

- Use `--` to comment a line:

```
SELECT 1+1;   -- comments
```

```
+------+
| 1+1 |
+------+
|   2 |
+------+
1 row in set (0.00 sec)
```

And this style requires at least one whitespace after `--`:

```
SELECT 1+1--1;
```

```
+--------+
| 1+1--1 |
+--------+
|      3 |
+--------+
1 row in set (0.01 sec)
```

- Use `/* */` to comment a block or multiple lines:

```
SELECT 1 /* this is an in-line comment */ + 1;
```

```
+--------+
| 1  + 1 |
+--------+
|      2 |
+--------+
1 row in set (0.01 sec)
```

```
SELECT 1+
    -> /*
/*> this is a
/*> multiple-line comment
/*> */
    -> 1;
```

```
+-------+
| 1+
1 |
+-------+
|    2 |
+-------+
1 row in set (0.00 sec)
```

4.1.2.6.1 MySQL-compatible comment syntax

The same as MySQL, TiDB supports a variant of C comment style:

```
/*! Specific code */
```

or

```
/*!50110 Specific code */
```

In this style, TiDB runs the statements in the comment.

For example:

```
SELECT /*! STRAIGHT_JOIN */ col1 FROM table1,table2 WHERE ...
```

In TiDB, you can also use another version:

```
SELECT STRAIGHT_JOIN col1 FROM table1,table2 WHERE ...
```

If the server version number is specified in the comment, for example, `/*!50110 KEY_BLOCK_SIZE=1024 */`, in MySQL it means that the contents in this comment are processed only when the MySQL version is or higher than 5.1.10. But in TiDB, the MySQL version number does not work and all contents in the comment are processed.

4.1.2.6.2 TiDB specific comment syntax

TiDB has its own comment syntax (that is, TiDB specific comment syntax) with the format of `/*T![feature_id] XXX */`. TiDB can parse the SQL fragment in this comment only if it implements the corresponding feature of `feature_id` in the current version. For example, as the `AUTO_RANDOM` feature is introduced in v3.1.1, this version of TiDB can parse `/*T![auto_rand] auto_random */` into `auto_random`. Because the `AUTO_RANDOM` feature is not implemented in v3.0.0, the SQL statement fragment above is ignored.

4.1.2.6.3 Optimizer comment syntax

Another type of comment is specially treated as an optimizer hint:

```
SELECT /*+ hint */ FROM ...;
```

For details about the optimizer hints that TiDB supports, see Optimizer hints.

> **Note**
>
> In MySQL client before 5.7.7, TiDB specific comment syntax and optimizer comment syntax are treated as comments and cleared by default. To use the two syntaxes in the old client, add the `--comments` option when you start the client. For example, `mysql -h 127.0.0.1 -P 4000 -uroot --comments`.

For more information, see Comment Syntax.

4.1.3 Data Types

4.1.3.1 Data Types

TiDB supports all the data types in MySQL except the `SPATIAL` type. This includes all the numeric types, string types, date & time types, and the JSON type.

The definitions used for datatypes are specified as `T(M[, D])`. Where by:

- `T` indicates the specific data type.
- `M` indicates the maximum display width for integer types. For floating-point and fixed-point types, `M` is the total number of digits that can be stored (the precision). For string types, `M` is the maximum length. The maximum permissible value of M depends on the data type.
- `D` applies to floating-point and fixed-point types and indicates the number of digits following the decimal point (the scale).
- `fsp` applies to the `TIME`, `DATETIME`, and `TIMESTAMP` types and represents the fractional seconds precision. The `fsp` value, if given, must be in the range 0 to 6. A value of 0 signifies that there is no fractional part. If omitted, the default precision is 0.

4.1.3.2 Default Values

The `DEFAULT` value clause in a data type specification indicates a default value for a column. The default value must be a constant and cannot be a function or an expression. But for the time type, you can specify the `NOW`, `CURRENT_TIMESTAMP`, `LOCALTIME`, and `LOCALTIMESTAMP` functions as the default for `TIMESTAMP` and `DATETIME` columns.

The `BLOB`, `TEXT`, and `JSON` columns **cannot** be assigned a default value.

If a column definition includes no explicit `DEFAULT` value, TiDB determines the default value as follows:

- If the column can take `NULL` as a value, the column is defined with an explicit `DEFAULT` ↪ `NULL` clause.

- If the column cannot take `NULL` as the value, TiDB defines the column with no explicit `DEFAULT` clause.

For data entry into a `NOT NULL` column that has no explicit `DEFAULT` clause, if an `INSERT` or `REPLACE` statement includes no value for the column, TiDB handles the column according to the SQL mode in effect at the time:

- If strict SQL mode is enabled, an error occurs for transactional tables, and the statement is rolled back. For nontransactional tables, an error occurs.
- If strict mode is not enabled, TiDB sets the column to the implicit default value for the column data type.

Implicit defaults are defined as follows:

- For numeric types, the default is 0. If declared with the `AUTO_INCREMENT` attribute, the default is the next value in the sequence.
- For date and time types other than `TIMESTAMP`, the default is the appropriate "zero" value for the type. For `TIMESTAMP`, the default value is the current date and time.
- For string types other than `ENUM`, the default value is the empty string. For `ENUM`, the default is the first enumeration value.

4.1.3.3 Numeric Types

4.1.3.3.1 Numeric Types

TiDB supports all the MySQL numeric types, including:

- Integer Types (Exact Value)
- Floating-Point Types (Approximate Value)
- Fixed-Point Types (Exact Value)

Integer types

TiDB supports all the MySQL integer types, including `INTEGER/INT`, `TINYINT`, `SMALLINT` ↪ , `MEDIUMINT`, and `BIGINT`. For more information, see Numeric Data Type Syntax in MySQL.

The following table summarizes field descriptions:

Syntax Element	Description
M	the display width of the type. Optional.
UNSIGNED	UNSIGNED. If omitted, it is SIGNED.

Syntax Element	Description
ZEROFILL	If you specify ZEROFILL for a numeric column, TiDB automatically adds the UNSIGNED attribute to the column.

The following table summarizes the required storage and range for integer types supported by TiDB:

Data Type	Storage Required (bytes)	Minimum Value (signed/unsigned)	Maximum value (signed/unsigned)
`TINYINT`	1	-128 / 0	127 / 255
`SMALLINT`	2	-32768 / 0	32767 / 65535
`MEDIUMINT`	3	-8388608 / 0	8388607 / 16777215
`INT`	4	-2147483648 / 0	2147483647 / 4294967295
`BIGINT`	8	-9223372036854775808 / 0	9223372036854775807 / 18446744073709551615

`BIT` type

The BIT data type. A type of BIT(M) enables the storage of M-bit values. M can range from 1 to 64, with the default value of 1:

```
BIT[(M)]
```

`BOOLEAN` type

The `BOOLEAN` type and its alias `BOOL` are equivalent to `TINYINT(1)`. If the value is `0`, it is considered as `False`; otherwise, it is considered `True`. As in MySQL, `True` is `1` and `False` is `0`:

```
BOOLEAN
```

`TINYINT` type

The TINYINT data type stores signed values of range [-128, 127] and unsigned values of range [0, 255]:

```
TINYINT[(M)] [UNSIGNED] [ZEROFILL]
```

SMALLINT type

The SMALLINT data type stores signed values of range [-32768, 32767], and unsigned values of range [0, 65535]:

```
SMALLINT[(M)] [UNSIGNED] [ZEROFILL]
```

MEDIUMINT type

The MEDIUMINT data type stores signed values of range [-8388608, 8388607], and unsigned values of range [0, 16777215]:

```
MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]
```

INTEGER type

The INTEGER type and its alias INT stores signed values of range [-2147483648, 2147483647], and unsigned values of range [0, 4294967295]:

```
INT[(M)] [UNSIGNED] [ZEROFILL]
```

You can also use another form:

```
INTEGER[(M)] [UNSIGNED] [ZEROFILL]
```

BIGINT type

The BIGINT data type stores signed values of range [-9223372036854775808, 9223372036854775807], and unsigned values of range [0, 18446744073709551615]:

```
BIGINT[(M)] [UNSIGNED] [ZEROFILL]
```

Floating-point types

TiDB supports all the MySQL floating-point types, including FLOAT, and DOUBLE. For more information, see Floating-Point Types (Approximate Value) - FLOAT, DOUBLE in MySQL.

The following table summarizes field descriptions:

Syntax Element	Description
M	the total number of digits
D	the number of digits following the decimal point
UNSIGNED	UNSIGNED. If omitted, it is SIGNED.

Syntax Element	Description
ZEROFILL	If you specify ZEROFILL for a numeric column, TiDB automatically adds the UNSIGNED attribute to the column.

The following table summarizes the required storage for floating-point types supported by TiDB:

Data Type	Storage Required (bytes)
`FLOAT`	4
`FLOAT(p)`	If 0 <= p <= 24, it is 4; if 25 <= p <= 53, it is 8
`DOUBLE`	8

`FLOAT` type

The `FLOAT` type stores a single-precision floating-point number. Permissible values are -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38. These are the theoretical limits, based on the IEEE standard. The actual range might be slightly smaller depending on your hardware or operating system.

`FLOAT(p)` can be used to represent the required precision in bits. TiDB uses this value only to determine whether to use `FLOAT` or `DOUBLE` for the resulting data type. If p is from 0 to 24, the data type becomes FLOAT with no M or D values. If p is from 25 to 53, the data type becomes `DOUBLE` with no M or D values. The range of the resulting column is the same as for the single-precision `FLOAT` or double-precision `DOUBLE` data type.

```
FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]
FLOAT(p) [UNSIGNED] [ZEROFILL]
```

Warning:

As in MySQL, the `FLOAT` data type stores approximate values. For values such as currency, it is recommended to use the `DECIMAL` type instead.

`DOUBLE` type

The `DOUBLE` type, and its alias `DOUBLE PRECISION` stores a double-precision floating-point number. Permissible values are -1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308. These are the theo-

retical limits, based on the IEEE standard. The actual range might be slightly smaller depending on your hardware or operating system.

```
DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]
DOUBLE PRECISION [(M,D)] [UNSIGNED] [ZEROFILL], REAL[(M,D)] [UNSIGNED] [
    ↪ ZEROFILL]
```

> **Warning:**
>
> As in MySQL, the `DOUBLE` data type stores approximate values. For values such as currency, it is recommended to use the `DECIMAL` type instead.

Fixed-point types

TiDB supports all the MySQL floating-point types, including DECIMAL, and NUMERIC. For more information, Fixed-Point Types (Exact Value) - DECIMAL, NUMERIC in MySQL.

The meaning of the fields:

Syntax Element	Description
M	the total number of digits
D	the number of digits after the decimal point
UNSIGNED	UNSIGNED. If omitted, it is SIGNED.
ZEROFILL	If you specify ZEROFILL for a numeric column, TiDB automatically adds the UNSIGNED attribute to the column.

`DECIMAL` type

`DECIMAL` and its alias `NUMERIC` stores a packed "exact" fixed-point number. M is the total number of digits (the precision), and D is the number of digits after the decimal point (the scale). The decimal point and (for negative numbers) the - sign are not counted in M. If D is 0, values have no decimal point or fractional part. The maximum number of digits (M) for DECIMAL is 65. The maximum number of supported decimals (D) is 30. If D is omitted, the default is 0. If M is omitted, the default is 10.

```
DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]
```

```
NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL]
```

4.1.3.4 Date and Time Types

4.1.3.4.1 Date and Time Types

TiDB supports all MySQL date and time data types to store temporal values: `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`, and `YEAR`. For more information, see Date and Time Data Types in MySQL.

Each of these types has its range of valid values, and uses a zero value to indicate that it is an invalid value. In addition, the `TIMESTAMP` and `DATETIME` types can automatically generate new time values on modification.

When dealing with date and time value types, note:

- Although TiDB tries to interpret different formats, the date-portion must be in the format of year-month-day (for example, '1998-09-04'), rather than month-day-year or day-month-year.
- If the year-portion of a date is specified as 2 digits, TiDB converts it based on specific rules.
- If a numeric value is needed in the context, TiDB automatically converts the date or time value into a numeric type. For example:

```
mysql> SELECT NOW(), NOW()+0, NOW(3)+0;
+---------------------+---------------+--------------------+
| NOW()               | NOW()+0       | NOW(3)+0           |
+---------------------+---------------+--------------------+
| 2012-08-15 09:28:00 | 20120815092800 | 20120815092800.889 |
+---------------------+---------------+--------------------+
```

- TiDB might automatically convert invalid values or values beyond the supported range to a zero value of that type. This behavior is dependent on the SQL Mode set. For example:

```
mysql> show create table t1;
+--
   ↪ -----+-----------------------------------------------------------------
   ↪
| Table | Create Table
   ↪
   ↪ |
+--
   ↪ -----+-----------------------------------------------------------------
   ↪
```

```
| t1    | CREATE TABLE `t1` (
  `a` time DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin |
+--
    ↪ -----+----------------------------------------------------------------
    ↪
1 row in set (0.00 sec)

mysql> select @@sql_mode;
+--
    ↪ ----------------------------------------------------------------------
    ↪
| @@sql_mode
    ↪
    ↪ |
+--
    ↪ ----------------------------------------------------------------------
    ↪
| ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,
    ↪ ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,
    ↪ NO_ENGINE_SUBSTITUTION |
+--
    ↪ ----------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)

mysql> insert into t1 values ('2090-11-32:22:33:44');
ERROR 1292 (22007): Truncated incorrect time value: '
    ↪ 2090-11-32:22:33:44'
mysql> set @@sql_mode='';
    ↪
    ↪ Query OK, 0 rows affected (0.01 sec)

mysql> insert into t1 values ('2090-11-32:22:33:44');
Query OK, 1 row affected, 1 warning (0.01 sec)

mysql> select * from t1;
+----------+
| a        |
+----------+
| 00:00:00 |
+----------+
1 row in set (0.01 sec)
```

- Setting different SQL modes can change TiDB behaviors.
- If the SQL mode `NO_ZERO_DATE` is not enabled, TiDB allows month or day in the columns of `DATE` and `DATETIME` to be zero value, for example, '2009-00-00' or '2009-01-00'. If this date type is to be calculated in a function, for example, in `DATE_SUB()` or `DATE_ADD()`, the result can be incorrect.
- By default, TiDB enables the SQL mode `NO_ZERO_DATE`. This mode prevents storing zero values such as '0000-00-00'.

Different types of zero value are shown in the following table:

Date Type	"Zero" Value
DATE	'0000-00-00'
TIME	'00:00:00'
DATETIME	'0000-00-00 00:00:00'
TIMESTAMP	'0000-00-00 00:00:00'
YEAR	0000

Invalid `DATE`, `DATETIME`, `TIMESTAMP` values are automatically converted to the corresponding type of zero value ('0000-00-00' or '0000-00-00 00:00:00') if the SQL mode permits such usage.

Supported types

`DATE` type

`DATE` only contains date-portion and no time-portion, displayed in `YYYY-MM-DD` format. The supported range is '1000-01-01' to '9999-12-31':

```
DATE
```

`TIME` type

For the `TIME` type, the format is `HH:MM:SS[.fraction]` and valid values range from '-838:59:59.000000' to '838:59:59.000000'. `TIME` is used not only to indicate the time within a day but also to indicate the time interval between 2 events. An optional `fsp` value in the range from 0 to 6 may be given to specify fractional seconds precision. If omitted, the default precision is 0:

```
TIME[(fsp)]
```

Note:

Pay attention to the abbreviated form of `TIME`. For example, '11:12' means '11:12:00' instead of '00:11:12'. However, '1112' means '00:11:12'. These differences are caused by the presence or absence of the `:` character.

`DATETIME` type

`DATETIME` contains both date-portion and time-portion. Valid values range from '1000-01-01 00:00:00.000000' to '9999-12-31 23:59:59.999999'.

TiDB displays `DATETIME` values in `YYYY-MM-DD HH:MM:SS[.fraction]` format, but permits assignment of values to `DATETIME` columns using either strings or numbers. An optional fsp value in the range from 0 to 6 may be given to specify fractional seconds precision. If omitted, the default precision is 0:

```
DATETIME[(fsp)]
```

`TIMESTAMP` type

`TIMESTAMP` contains both date-portion and time-portion. Valid values range from '1970-01-01 00:00:01.000000' to '2038-01-19 03:14:07.999999' in UTC time. An optional fsp value in the range from 0 to 6 may be given to specify fractional seconds precision. If omitted, the default precision is 0.

In `TIMESTAMP`, zero is not permitted to appear in the month-portion or day-portion. The only exception is zero value itself '0000-00-00 00:00:00'.

```
TIMESTAMP[(fsp)]
```

Timezone Handling

When `TIMESTAMP` is to be stored, TiDB converts the `TIMESTAMP` value from the current time zone to UTC time zone. When `TIMESTAMP` is to be retrieved, TiDB converts the stored `TIMESTAMP` value from UTC time zone to the current time zone (Note: `DATETIME` is not handled in this way). The default time zone for each connection is the server's local time zone, which can be modified by the environment variable `time_zone`.

Warning:

As in MySQL, the `TIMESTAMP` data type suffers from the Year 2038 Problem. For storing values that may span beyond 2038, please consider using the `DATETIME` type instead.

`YEAR` type

The `YEAR` type is specified in the format 'YYYY'. Supported values range from 1901 to 2155, or the zero value of 0000:

```
YEAR[(4)]
```

`YEAR` follows the following format rules:

- Four-digit numeral ranges from 1901 to 2155
- Four-digit string ranges from '1901' to '2155'
- One-digit or two-digit numeral ranges from 1 to 99. Accordingly, 1-69 is converted to 2001-2069 and 70-99 is converted to 1970-1999
- One-digit or two-digit string ranges from '0' to '99'
- Value 0 is taken as 0000 whereas the string '0' or '00' is taken as 2000

Invalid `YEAR` value is automatically converted to 0000 (if users are not using the `NO_ZERO_DATE` SQL mode).

Automatic initialization and update of `TIMESTAMP` and `DATETIME`

Columns with `TIMESTAMP` or `DATETIME` value type can be automatically initialized or updated to the current time.

For any column with `TIMESTAMP` or `DATETIME` value type in the table, you can set the default or auto-update value as current timestamp.

These properties can be set by setting `DEFAULT CURRENT_TIMESTAMP` and `ON UPDATE CURRENT_TIMESTAMP` when the column is being defined. DEFAULT can also be set as a specific value, such as `DEFAULT 0` or `DEFAULT '2000-01-01 00:00:00'`.

```
CREATE TABLE t1 (
    ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    dt DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

The default value for `DATETIME` is `NULL` unless it is specified as `NOT NULL`. For the latter situation, if no default value is set, the default value is be 0.

```
CREATE TABLE t1 (
    dt1 DATETIME ON UPDATE CURRENT_TIMESTAMP,   -- default NULL
    dt2 DATETIME NOT NULL ON UPDATE CURRENT_TIMESTAMP -- default 0
);
```

Decimal part of time value

`DATETIME` and `TIMESTAMP` values can contain a fractional part of up to 6 digits which is accurate to milliseconds. In any column of `DATETIME` or `TIMESTAMP` types, a fractional part is stored instead of being discarded. With a fractional part, the value is in the format of 'YYYY-MM-DD HH:MM:SS[.fraction]', and the fraction ranges from 000000 to 999999. A decimal point must be used to separate the fraction from the rest.

- Use `type_name(fsp)` to define a column that supports fractional precision, where `type_name` can be `TIME`, `DATETIME` or `TIMESTAMP`. For example,

  ```
  CREATE TABLE t1 (t TIME(3), dt DATETIME(6));
  ```

`fsp` must range from 0 to 6.

`0` means there is no fractional part. If `fsp` is omitted, the default is 0.

- When inserting `TIME`, `DATETIME` or `TIMESTAMP` which contain a fractional part, if the number of digit of the fraction is too few, or too many, rounding might be needed in the situation. For example:

```
mysql> CREATE TABLE fractest( c1 TIME(2), c2 DATETIME(2), c3 TIMESTAMP
    ↪ (2) );
Query OK, 0 rows affected (0.33 sec)

mysql> INSERT INTO fractest VALUES
    > ('17:51:04.777', '2014-09-08 17:51:04.777', '2014-09-08
       ↪ 17:51:04.777');
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM fractest;
+-------------/-----------------------/-----------------------+
| c1          | c2                    | c3                    |
+-------------/-----------------------/-----------------------+
| 17:51:04.78 | 2014-09-08 17:51:04.78 | 2014-09-08 17:51:04.78 |
+-------------/-----------------------/-----------------------+
1 row in set (0.00 sec)
```

Conversions between date and time types

Sometimes we need to make conversions between date and time types. But some conversions might lead to information loss. For example, `DATE`, `DATETIME` and `TIMESTAMP` values all have their own respective ranges. `TIMESTAMP` should be no earlier than the year 1970 in UTC time or no later than UTC time '2038-01-19 03:14:07'. Based on this rule, '1968-01-01' is a valid date value of `DATE` or `DATETIME`, but becomes 0 when it is converted to `TIMESTAMP`.

The conversions of `DATE`:

- When `DATE` is converted to `DATETIME` or `TIMESTAMP`, a time-portion '00:00:00' is added, because DATE does not contain any time information
- When `DATE` is converted to `TIME`, the result is '00:00:00'

Conversions of `DATETIME` or `TIMESTAMP`:

- When `DATETIME` or `TIMESTAMP` is converted to `DATE`, the time and fractional part is discarded. For example, '1999-12-31 23:59:59.499' is converted to '1999-12-31'
- When `DATETIME` or `TIMESTAMP` is converted to TIME, the time-portion is discarded, because `TIME` does not contain any time information

When we convert `TIME` to other time and date formats, the date-portion is automatically specified as `CURRENT_DATE()`. The final converted result is a date that consists of `TIME` and `CURRENT_DATE()`. This is to say that if the value of TIME is beyond the range from '00:00:00' to '23:59:59', the converted date-portion does not indicate the current day.

When `TIME` is converted to `DATE`, the process is similar, and the time-portion is discarded.

Using the `CAST()` function can explicitly convert a value to a `DATE` type. For example:

```
date_col = CAST(datetime_col AS DATE)
```

Converting `TIME` and `DATETIME` to numeric format. For example:

```
mysql> SELECT CURTIME(), CURTIME()+0, CURTIME(3)+0;
+-----------/-------------/--------------+
| CURTIME() | CURTIME()+0 | CURTIME(3)+0 |
+-----------/-------------/--------------+
| 09:28:00 |      92800 |   92800.887 |
+-----------/-------------/--------------+
mysql> SELECT NOW(), NOW()+0, NOW(3)+0;
+---------------------/----------------/--------------------+
| NOW()               | NOW()+0       | NOW(3)+0          |
+---------------------/----------------/--------------------+
| 2012-08-15 09:28:00 | 20120815092800 | 20120815092800.889 |
+---------------------/----------------/--------------------+
```

Two-digit year-portion contained in the date

The two-digit year-portion contained in date does not explicitly indicate the actual year and is ambiguous.

For `DATETIME`, `DATE` and `TIMESTAMP` types, TiDB follows the following rules to eliminate ambiguity:

- Values between 01 and 69 is converted to a value between 2001 and 2069
- Values between 70 and 99 is converted to a value between 1970 and 1999

These rules also apply to the `YEAR` type, with one exception:

When numeral `00` is inserted to `YEAR(4)`, the result is 0000 rather than 2000.

If you want the result to be 2000, specify the value to be 2000.

The two-digit year-portion might not be properly calculated in some functions such `MIN()` and `MAX()`. For these functions, the four-digit format suites better.

4.1.3.5 String Types

4.1.3.5.1 String Types

TiDB supports all the MySQL string types, including `CHAR`, `VARCHAR`, `BINARY`, `VARBINARY` ↪ , `BLOB`, `TEXT`, `ENUM`, and `SET`. For more information, see String Types in MySQL.

Supported types

`CHAR` type

`CHAR` is a fixed length string. Values stored as `CHAR` are right-padded with spaces to the specified length. M represents the column-length in characters (not bytes). The range of M is 0 to 255:

```
[NATIONAL] CHAR[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]
```

`VARCHAR` type

`VARCHAR` is a string of variable-length. M represents the maximum column length in characters (not bytes). The maximum size of `VARCHAR` cannot exceed 65,535 bytes. The maximum row length and the character set being used determine the `VARCHAR` length.

The space occupied by a single character might differ for different character sets. The following table shows the bytes consumed by a single character, and the range of the `VARCHAR` column length in each character set:

Character Set	Byte(s) per Character	Range of the Maximum `VARCHAR` Column Length
ascii	1	(0, 65535]
latin1	1	(0, 65535]
binary	1	(0, 65535]
utf8	3	(0, 21845]
utf8mb4	4	(0, 16383]

```
[NATIONAL] VARCHAR(M) [CHARACTER SET charset_name] [COLLATE collation_name]
```

`TEXT` type

`TEXT` is a string of variable-length. M represents the maximum column length in characters, ranging from 0 to 65,535. The maximum row length and the character set being used determine the `TEXT` length.

```
TEXT[(M)] [CHARACTER SET charset_name] [COLLATE collation_name]
```

`TINYTEXT` type

The `TINYTEXT` type is similar to the TEXT type. The difference is that the maximum column length of `TINYTEXT` is 255.

```
TINYTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

`MEDIUMTEXT` type

The MEDIUMTEXT type is similar to the TEXT type. The difference is that the maximum column length of MEDIUMTEXT is 16,777,215.

```
MEDIUMTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

LONGTEXT type

The LONGTEXT type is similar to the TEXT type. The difference is that the maximum column length of LONGTEXT is 4,294,967,295.

```
LONGTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
```

BINARY type

The BINARY type is similar to the CHAR type. The difference is that BINARY stores binary byte strings.

```
BINARY(M)
```

VARBINARY type

The VARBINARY type is similar to the VARCHAR type. The difference is that the VARBINARY stores binary byte strings.

```
VARBINARY(M)
```

BLOB type

BLOB is a large binary file. M represents the maximum column length in bytes, ranging from 0 to 65,535.

```
BLOB[(M)]
```

TINYBLOB type

The TINYBLOB type is similar to the BLOB type. The difference is that the maximum column length of TINYBLOB is 255.

```
TINYBLOB
```

MEDIUMBLOB type

The MEDIUMBLOB type is similar to the BLOB type. The difference is that the maximum column length of MEDIUMBLOB is 16,777,215.

```
MEDIUMBLOB
```

LONGBLOB type

The LONGBLOB type is similar to the BLOB type. The difference is that the maximum column length of LONGBLOB is 4,294,967,295.

```
LONGBLOB
```

`ENUM` type

An `ENUM` is a string object with a value chosen from a list of permitted values that are enumerated explicitly in the column specification when the table is created. The syntax is:

```
ENUM('value1','value2',...) [CHARACTER SET charset_name] [COLLATE
    ↪ collation_name]

##### For example:
ENUM('apple', 'orange', 'pear')
```

The value of the `ENUM` data type is stored as numbers. Each value is converted to a number according the definition order. In the previous example, each string is mapped to a number:

Value	Number
NULL	NULL
”	0
‘apple’	1
‘orange’	2
‘pear’	3

For more information, see the ENUM type in MySQL.

`SET` type

A `SET` is a string object that can have zero or more values, each of which must be chosen from a list of permitted values specified when the table is created. The syntax is:

```
SET('value1','value2',...) [CHARACTER SET charset_name] [COLLATE
    ↪ collation_name]

##### For example:
SET('1', '2') NOT NULL
```

In the example, any of the following values can be valid:

```
''
'1'
'2'
'1,2'
```

In TiDB, the values of the `SET` type is internally converted to `Int64`. The existence of each element is represented using a binary: 0 or 1. For a column specified as `SET('a','b ↪ ','c','d')`, the members have the following decimal and binary values.

Member	Decimal Value	Binary Value
'a'	1	0001
'b'	2	0010
'c'	4	0100
'd'	8	1000

In this case, for an element of `('a', 'c')`, it is `0101` in binary.

For more information, see the SET type in MySQL.

4.1.3.6 JSON Type

> **Warning:**
>
> This is still an experimental feature. It is recommended **not** to use this feature in the production environment.

TiDB supports the `JSON` (JavaScript Object Notation) data type, which is useful for storing semi-structured data. The `JSON` data type provides the following advantages over storing `JSON`-format strings in a string column:

- Use the Binary format for serialization. The internal format permits quick read access to `JSON` document elements.
- Automatic validation of the JSON documents stored in `JSON` columns. Only valid documents can be stored.

`JSON` columns, like columns of other binary types, are not indexed directly, but you can index the fields in the `JSON` document in the form of generated column:

```
CREATE TABLE city (
    id INT PRIMARY KEY,
    detail JSON,
    population INT AS (JSON_EXTRACT(detail, '$.population')),
    index index_name (population)
    );
INSERT INTO city (id,detail) VALUES (1, '{"name": "Beijing", "population":
    ↪ 100}');
SELECT id FROM city WHERE population >= 100;
```

For more information, see JSON Functions and Generated Columns.

4.1.4 Functions and Operators

4.1.4.1 Function and Operator Reference

The usage of the functions and operators in TiDB is similar to MySQL. See Functions and Operators in MySQL.

In SQL statements, expressions can be used on the `ORDER BY` and `HAVING` clauses of the `SELECT` statement, the `WHERE` clause of `SELECT`/`DELETE`/`UPDATE` statements, and `SET` statements.

You can write expressions using literals, column names, NULL, built-in functions, operators and so on. For expressions that TiDB supports pushing down to TiKV, see List of Expressions for Pushdown.

4.1.4.2 Type Conversion in Expression Evaluation

TiDB behaves the same as MySQL: https://dev.mysql.com/doc/refman/5.7/en/type-conversion.html

4.1.4.3 Operators

This document describes the operators precedence, comparison functions and operators, logical operators, and assignment operators.

- Operator precedence
- Comparison functions and operators
- Logical operators
- Assignment operators

Name	Description
AND, &&	Logical AND
=	Assign a value (as part of a `SET` statement, or as part of the `SET` clause in an `UPDATE` statement)
:=	Assign a value
BETWEEN … AND …	Check whether a value is within a range of values
BINARY	Cast a string to a binary string
&	Bitwise AND
~	Bitwise inversion
|	Bitwise OR
[^](https://dev.mysql.com/doc/refman/5.7/en/bit-functions.html#operator_bitwise-xor)	Bitwise XOR
CASE	Case operator
DIV	Integer division

Name	Description
/	Division operator
=	Equal operator
<=>	NULL-safe equal to operator
>	Greater than operator
>=	Greater than or equal operator
IS	Test a value against a boolean
IS NOT	Test a value against a boolean
IS NOT NULL	NOT NULL value test
IS NULL	NULL value test
->	Return value from JSON column after evaluating path; equivalent to `JSON_EXTRACT()`
->>	Return value from JSON column after evaluating path and unquoting the result; equivalent to `JSON_UNQUOTE(JSON_EXTRACT())`
<<	Left shift
<	Less than operator
<=	Less than or equal operator
LIKE	Simple pattern matching
-	Minus operator
%, MOD	Modulo operator
NOT, !	Negates value
NOT BETWEEN ... AND ...	Check whether a value is not within a range of values
!=, <>	Not equal operator
NOT LIKE	Negation of simple pattern matching
NOT REGEXP	Negation of REGEXP
||, OR	Logical OR
+	Addition operator
REGEXP	Pattern matching using regular expressions
>>	Right shift
RLIKE	Synonym for REGEXP
SOUNDS LIKE	Compare sounds
*	Multiplication operator
-	Change the sign of the argument
XOR	Logical XOR

4.1.4.3.1 Operator precedence

Operator precedences are shown in the following list, from highest precedence to the lowest. Operators that are shown together on a line have the same precedence.

```
INTERVAL
```

```
BINARY, COLLATE
!
- (unary minus), ~ (unary bit inversion)
^
*, /, DIV, %, MOD
-, +
<<, >>
&
|
= (comparison), <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
BETWEEN, CASE, WHEN, THEN, ELSE
NOT
AND, &&
XOR
OR, ||
= (assignment), :=
```

For details, see Operator Precedence.

4.1.4.3.2 Comparison functions and operators

Name	Description
BETWEEN … AND …	Check whether a value is within a range of values
COALESCE()	Return the first non-NULL argument
=	Equal operator
<=>	NULL-safe equal to operator
>	Greater than operator
>=	Greater than or equal operator
GREATEST()	Return the largest argument
IN()	Check whether a value is within a set of values
INTERVAL()	Return the index of the argument that is less than the first argument
IS	Test a value against a boolean
IS NOT	Test a value against a boolean
IS NOT NULL	NOT NULL value test
IS NULL	NULL value test
ISNULL()	Test whether the argument is NULL
LEAST()	Return the smallest argument
<	Less than operator
<=	Less than or equal operator
LIKE	Simple pattern matching
NOT BETWEEN … AND …	Check whether a value is not within a range of values
!=, <>	Not equal operator
NOT IN()	Check whether a value is not within a set of values
NOT LIKE	Negation of simple pattern matching

Name	Description
STRCMP()	Compare two strings

For details, see Comparison Functions and Operators.

4.1.4.3.3 Logical operators

Name	Description
AND, &&	Logical AND
NOT, !	Negates value
||, OR	Logical OR
XOR	Logical XOR

For details, see MySQL Handling of GROUP BY.

4.1.4.3.4 Assignment operators

Name	Description
=	Assign a value (as part of a `SET` statement, or as part of the `SET` clause in an `UPDATE` statement)
:=	Assign a value

For details, see Detection of Functional Dependence.

4.1.4.4 Control Flow Functions

TiDB supports all of the control flow functions available in MySQL 5.7.

Name	Description
`CASE`	Case operator
`IF()`	If/else construct
`IFNULL()`	Null if/else construct
`NULLIF()`	Return NULL if expr1 = expr2

4.1.4.5 String Functions

TiDB supports most of the string functions available in MySQL 5.7.

4.1.4.5.1 Supported functions

Name	Description
ASCII()	Return numeric value of left-most character
BIN()	Return a string containing binary representation of a number
BIT_LENGTH()	Return length of argument in bits
CHAR()	Return the character for each integer passed
CHAR_LENGTH()	Return number of characters in argument
CHARACTER_LENGTH()	Synonym for CHAR_LENGTH()
CONCAT()	Return concatenated string
CONCAT_WS()	Return concatenate with separator
ELT()	Return string at index number
EXPORT_SET()	Return a string such that for every bit set in the value bits, you get an on string and for every unset bit, you get an off string
FIELD()	Return the index (position) of the first argument in the subsequent arguments
FIND_IN_SET()	Return the index position of the first argument within the second argument
FORMAT()	Return a number formatted to specified number of decimal places
FROM_BASE64()	Decode to a base-64 string and return result
HEX()	Return a hexadecimal representation of a decimal or string value
INSERT()	Insert a substring at the specified position up to the specified number of characters
INSTR()	Return the index of the first occurrence of substring
LCASE()	Synonym for LOWER()
LEFT()	Return the leftmost number of characters as specified
LENGTH()	Return the length of a string in bytes
LIKE	Simple pattern matching
LOCATE()	Return the position of the first occurrence of substring
LOWER()	Return the argument in lowercase
LPAD()	Return the string argument, left-padded with the specified string
LTRIM()	Remove leading spaces
MAKE_SET()	Return a set of comma-separated strings that have the corresponding bit in bits set
MID()	Return a substring starting from the specified position
NOT LIKE	Negation of simple pattern matching
NOT REGEXP	Negation of REGEXP

Name	Description
OCT()	Return a string containing octal representation of a number
OCTET_LENGTH()	Synonym for LENGTH()
ORD()	Return character code for leftmost character of the argument
POSITION()	Synonym for LOCATE()
QUOTE()	Escape the argument for use in an SQL statement
REGEXP	Pattern matching using regular expressions
REPEAT()	Repeat a string the specified number of times
REPLACE()	Replace occurrences of a specified string
REVERSE()	Reverse the characters in a string
RIGHT()	Return the specified rightmost number of characters
RLIKE	Synonym for REGEXP
RPAD()	Append string the specified number of times
RTRIM()	Remove trailing spaces
SPACE()	Return a string of the specified number of spaces
STRCMP()	Compare two strings
SUBSTR()	Return the substring as specified
SUBSTRING()	Return the substring as specified
SUBSTRING_INDEX()	Return a substring from a string before the specified number of occurrences of the delimiter
TO_BASE64()	Return the argument converted to a base-64 string
TRIM()	Remove leading and trailing spaces
UCASE()	Synonym for UPPER()
UNHEX()	Return a string containing hex representation of a number
UPPER()	Convert to uppercase

4.1.4.5.2 Unsupported functions

- LOAD_FILE()
- MATCH
- SOUNDEX()
- SOUNDS LIKE
- WEIGHT_STRING()

4.1.4.6 Numeric Functions and Operators

TiDB supports all of the numeric functions and operators available in MySQL 5.7.

4.1.4.6.1 Arithmetic operators

Name	Description
+	Addition operator
-	Minus operator
*	Multiplication operator
/	Division operator
DIV	Integer division
lstinlineMOD!	Modulo operator
-	Change the sign of the argument

4.1.4.6.2 Mathematical functions

Name	Description
POW()	Return the argument raised to the specified power
POWER()	Return the argument raised to the specified power
EXP()	Raise to the power of
SQRT()	Return the square root of the argument
LN()	Return the natural logarithm of the argument
LOG()	Return the natural logarithm of the first argument
LOG2()	Return the base-2 logarithm of the argument
LOG10()	Return the base-10 logarithm of the argument
PI()	Return the value of pi
TAN()	Return the tangent of the argument
COT()	Return the cotangent
SIN()	Return the sine of the argument
COS()	Return the cosine
ATAN()	Return the arc tangent
ATAN2(), ATAN()	Return the arc tangent of the two arguments
ASIN()	Return the arc sine
ACOS()	Return the arc cosine
RADIANS()	Return argument converted to radians
DEGREES()	Convert radians to degrees
MOD()	Return the remainder
ABS()	Return the absolute value
CEIL()	Return the smallest integer value not less than the argument
CEILING()	Return the smallest integer value not less than the argument
FLOOR()	Return the largest integer value not greater than the argument
ROUND()	Round the argument
RAND()	Return a random floating-point value
SIGN()	Return the sign of the argument

Name	Description
CONV()	Convert numbers between different number bases
TRUNCATE()	Truncate to specified number of decimal places
CRC32()	Compute a cyclic redundancy check value

4.1.4.7 Date and Time Functions

TiDB supports all of the date and time functions available in MySQL 5.7.

Date/Time functions

Name	Description
ADDDATE()	Add time values (intervals) to a date value
ADDTIME()	Add time
CONVERT_TZ()	Convert from one time zone to another
CURDATE()	Return the current date
CURRENT_DATE(), CURRENT_DATE	Synonyms for CURDATE()
CURRENT_TIME(), CURRENT_TIME	Synonyms for CURTIME()
CURRENT_TIMESTAMP(), CURRENT_TIMESTAMP	Synonyms for NOW()
CURTIME()	Return the current time
DATE()	Extract the date part of a date or datetime expression
DATE_ADD()	Add time values (intervals) to a date value
DATE_FORMAT()	Format date as specified
DATE_SUB()	Subtract a time value (interval) from a date
DATEDIFF()	Subtract two dates
DAY()	Synonym for DAYOFMONTH()
DAYNAME()	Return the name of the weekday
DAYOFMONTH()	Return the day of the month (0-31)
DAYOFWEEK()	Return the weekday index of the argument
DAYOFYEAR()	Return the day of the year (1-366)
EXTRACT()	Extract part of a date
FROM_DAYS()	Convert a day number to a date
FROM_UNIXTIME()	Format Unix timestamp as a date
GET_FORMAT()	Return a date format string
HOUR()	Extract the hour
LAST_DAY	Return the last day of the month for the argument
LOCALTIME(), LOCALTIME	Synonym for NOW()
LOCALTIMESTAMP, LOCALTIMESTAMP()	Synonym for NOW()
MAKEDATE()	Create a date from the year and day of year

Name	Description
MAKETIME()	Create time from hour, minute, second
MICROSECOND()	Return the microseconds from argument
MINUTE()	Return the minute from the argument
MONTH()	Return the month from the date passed
MONTHNAME()	Return the name of the month
NOW()	Return the current date and time
PERIOD_ADD()	Add a period to a year-month
PERIOD_DIFF()	Return the number of months between periods
QUARTER()	Return the quarter from a date argument
SEC_TO_TIME()	Converts seconds to 'HH:MM:SS' format
SECOND()	Return the second (0-59)
STR_TO_DATE()	Convert a string to a date
SUBDATE()	Synonym for DATE_SUB() when invoked with three arguments
SUBTIME()	Subtract times
SYSDATE()	Return the time at which the function executes
TIME()	Extract the time portion of the expression passed
TIME_FORMAT()	Format as time
TIME_TO_SEC()	Return the argument converted to seconds
TIMEDIFF()	Subtract time
TIMESTAMP()	With a single argument, this function returns the date or datetime expression; with two arguments, the sum of the arguments
TIMESTAMPADD()	Add an interval to a datetime expression
TIMESTAMPDIFF()	Subtract an interval from a datetime expression
TO_DAYS()	Return the date argument converted to days
TO_SECONDS()	Return the date or datetime argument converted to seconds since Year 0
UNIX_TIMESTAMP()	Return a Unix timestamp
UTC_DATE()	Return the current UTC date
UTC_TIME()	Return the current UTC time
UTC_TIMESTAMP()	Return the current UTC date and time
WEEK()	Return the week number
WEEKDAY()	Return the weekday index
WEEKOFYEAR()	Return the calendar week of the date (1-53)
YEAR()	Return the year

Name	Description
YEARWEEK()	Return the year and week

For details, see Date and Time Functions.

4.1.4.8 Bit Functions and Operators

TiDB supports all of the bit functions and operators available in MySQL 5.7.

Bit functions and operators

Name	Description
BIT_COUNT()	Return the number of bits that are set as 1
&	Bitwise AND
~	Bitwise inversion
|	Bitwise OR
^	Bitwise XOR
<<	Left shift
>>	Right shift

[^](https://dev.mysql.com/doc/refman/5.7/en/bit-functions.html#operator_bitwise-xor)

4.1.4.9 Cast Functions and Operators

TiDB supports all of the cast functions and operators available in MySQL 5.7.

Name	Description
BINARY	Cast a string to a binary string
CAST()	Cast a value as a certain type
CONVERT()	Cast a value as a certain type

Cast functions and operators enable conversion of values from one data type to another.

4.1.4.10 Encryption and Compression Functions

TiDB supports most of the encryption and compression functions available in MySQL 5.7.

4.1.4.10.1 Supported functions

Name	Description
MD5()	Calculate MD5 checksum
PASSWORD()	Calculate and return a password string
RANDOM_BYTES()	Return a random byte vector
SHA1(), SHA()	Calculate an SHA-1 160-bit checksum
SHA2()	Calculate an SHA-2 checksum
AES_DECRYPT()	Decrypt using AES
AES_ENCRYPT()	Encrypt using AES
COMPRESS()	Return result as a binary string
UNCOMPRESS()	Uncompress a string compressed
UNCOMPRESSED_LENGTH()	Return the length of a string before compression
CREATE_ASYMMETRIC_PRIV_KEY()	Create private key
CREATE_ASYMMETRIC_PUB_KEY()	Create public key
CREATE_DH_PARAMETERS()	Generate shared DH secret
CREATE_DIGEST()	Generate digest from string
ASYMMETRIC_DECRYPT()	Decrypt ciphertext using private or public key
ASYMMETRIC_DERIVE()	Derive symmetric key from asymmetric keys
ASYMMETRIC_ENCRYPT()	Encrypt cleartext using private or public key
ASYMMETRIC_SIGN()	Generate signature from digest
ASYMMETRIC_VERIFY()	Verify that signature matches digest

4.1.4.10.2 Unsupported functions

- DES_DECRYPT(), DES_ENCRYPT(), OLD_PASSWORD(), ENCRYPT(): these functions were deprecated in MySQL 5.7 and removed in 8.0.
- VALIDATE_PASSWORD_STRENGTH()
- Functions only available in MySQL Enterprise Issue #2632

4.1.4.11 Information Functions

TiDB supports most of the information functions available in MySQL 5.7.

4.1.4.11.1 Supported functions

Name	Description
BENCHMARK()	Execute an expression in a loop
CONNECTION_ID()	Return the connection ID (thread ID) for the connection

Name	Description
CURRENT_USER()	Return the authenticated user name and host name
DATABASE()	Return the default (current) database name
FOUND_ROWS()	For a SELECT with a LIMIT clause, the number of the rows that are returned if there is no LIMIT clause
LAST_INSERT_ID()	Return the value of the AUTOINCREMENT column for the last INSERT
ROW_COUNT()	The number of rows affected
SCHEMA()	Synonym for DATABASE()
SESSION_USER()	Synonym for USER()
SYSTEM_USER()	Synonym for USER()
USER()	Return the user name and host name provided by the client
VERSION()	Return a string that indicates the MySQL server version

Name	Description
`TIDB_VERSION ↪ ()`	Return a string that indicates the TiDB server version

4.1.4.11.2 Unsupported functions

- `CHARSET()`
- `COERCIBILITY()`
- `COLLATION()`

4.1.4.12 JSON Functions

> **Warning:**
>
> This is still an experimental feature. It is recommended **not** to use this feature in the production environment.

TiDB supports most of the JSON functions that shipped with the GA release of MySQL 5.7. Additional JSON functions were added to MySQL 5.7 after its release, and not all are available in TiDB (see unsupported functions).

4.1.4.12.1 Functions that create JSON values

Function Name and Syntactic Sugar	Description
JSON_ARRAY([val[, val] ...])	Evaluates a (possibly empty) list of values and returns a JSON array containing those values

Function Name and Syntactic Sugar	Description
JSON_OBJECT(key, val[, key, val] …)	Evaluates a (possibly empty) list of key-value pairs and returns a JSON object containing those pairs
JSON_QUOTE(string)	Returns a string as a JSON value with quotes

4.1.4.12.2 Functions that search JSON values

Function Name and Syntactic Sugar	Description
JSON_CONTAINS(target, candidate[, path])	Indicates by returning 1 or 0 whether a given candidate JSON document is contained within a target JSON document
JSON_CONTAINS_PATH(json_doc, one_or_all, path[, path] …)	Returns 0 or 1 to indicate whether a JSON document contains data at a given path or paths

Function Name and Syntactic Sugar	Description
JSON_EXTRACT(json_doc, path[, path] …)	Returns data from a JSON document, selected from the parts of the document matched by the `path` arguments
->	Returns the value from a JSON column after the evaluating path; the syntactic sugar of `JSON_EXTRACT` ↪ `(doc,` ↪ `path_literal` ↪ `)`
->>	Returns the value from a JSON column after the evaluating path and unquoting the result; the syntactic sugar of `JSON_UNQUOTE` ↪ `(` ↪ `JSON_EXTRACT` ↪ `(doc,` ↪ `path_literal` ↪ `))`

Function Name and Syntactic Sugar	Description
JSON_KEYS(json_doc[, path])	Returns the keys from the top-level value of a JSON object as a JSON array, or, if a path argument is given, the top-level keys from the selected path
JSON_SEARCH(json_doc, one_or_all, search_string)	Search a JSON document for one or all matches of a string

4.1.4.12.3 Functions that modify JSON values

Function Name and Syntactic Sugar	Description
JSON_APPEND(json_doc, path, value)	An alias to `JSON_ARRAY_APPEND` ↪
JSON_ARRAY_APPEND(json_doc, path, value)	Appends a value to the end of a JSON array at a specified path
JSON_INSERT(json_doc, path, val[, path, val] ...)	Inserts data into a JSON document and returns the result

Function Name and Syntactic Sugar	Description
JSON_MERGE(json_doc, json_doc[, json_doc] …)	A deprecated alias for `JSON_MERGE_PRESERVE` ↪
JSON_MERGE_PRESERVE(json_doc, json_doc[, json_doc] …)	Merges two or more JSON documents and returns the merged result
JSON_REMOVE(json_doc, path[, path] …)	Removes data from a JSON document and returns the result
JSON_REPLACE(json_doc, path, val[, path, val] …)	Replaces existing values in a JSON document and returns the result
JSON_SET(json_doc, path, val[, path, val] …)	Inserts or updates data in a JSON document and returns the result
JSON_UNQUOTE(json_val)	Unquotes a JSON value and returns the result as a string

4.1.4.12.4 Functions that return JSON value attributes

Function Name and Syntactic Sugar	Description
JSON_DEPTH(json_doc)	Returns the maximum depth of a JSON document
JSON_LENGTH(json_doc[, path])	Returns the length of a JSON document, or, if a path argument is given, the length of the value within the path
JSON_TYPE(json_val)	Returns a string indicating the type of a JSON value

4.1.4.12.5 Unsupported functions

The following JSON functions are unsupported in TiDB. You can track the progress in adding them in TiDB #7546:

- `JSON_ARRAY_INSERT`
- `JSON_MERGE_PATCH`
- `JSON_PRETTY`
- `JSON_STORAGE_SIZE`
- `JSON_VALID`
- `JSON_ARRAYAGG`
- `JSON_OBJECTAGG`

4.1.4.13 Aggregate (GROUP BY) Functions

This document describes details about the supported aggregate functions in TiDB.

4.1.4.13.1 Supported aggregate functions

This section describes the supported MySQL `GROUP BY` aggregate functions in TiDB.

Name	Description
COUNT()	Return a count of the number of rows returned
COUNT(DISTINCT)	Return the count of a number of different values
SUM()	Return the sum
AVG()	Return the average value of the argument
MAX()	Return the maximum value
MIN()	Return the minimum value
GROUP_CONCAT()	Return a concatenated string
VARIANCE(), VAR_POP()	Return the population standard variance
STD(), STDDEV(), STDDEV_POP	Return the population standard deviation
VAR_SAMP()	Return the sample variance
STDDEV_SAMP()	Return the sample standard deviation

- Unless otherwise stated, group functions ignore NULL values.
- If you use a group function in a statement containing no GROUP BY clause, it is equivalent to grouping on all rows.

4.1.4.13.2 GROUP BY modifiers

TiDB does not currently support GROUP BY modifiers such as WITH ROLLUP. We plan to add support in the future. See TiDB #4250.

4.1.4.13.3 SQL mode support

TiDB supports the SQL Mode ONLY_FULL_GROUP_BY, and when enabled TiDB will refuse queries with ambiguous non-aggregated columns. For example, this query is illegal with ONLY_FULL_GROUP_BY enabled because the non-aggregated column "b" in the SELECT list does not appear in the GROUP BY statement:

```
drop table if exists t;
create table t(a bigint, b bigint, c bigint);
insert into t values(1, 2, 3), (2, 2, 3), (3, 2, 3);

mysql> select a, b, sum(c) from t group by a;
+------+------+--------+
| a    | b    | sum(c) |
+------+------+--------+
|    1 |    2 |      3 |
|    2 |    2 |      3 |
|    3 |    2 |      3 |
+------+------+--------+
3 rows in set (0.01 sec)

mysql> set sql_mode = 'ONLY_FULL_GROUP_BY';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select a, b, sum(c) from t group by a;
ERROR 1055 (42000): Expression #2 of SELECT list is not in GROUP BY clause
    ↪ and contains nonaggregated column 'b' which is not functionally
    ↪ dependent on columns in GROUP BY clause; this is incompatible with
    ↪ sql_mode=only_full_group_by
```

TiDB currently enables the ONLY_FULL_GROUP_BY mode by default.

Differences from MySQL

The current implementation of ONLY_FULL_GROUP_BY is less strict than that in MySQL 5.7. For example, suppose that we execute the following query, expecting the results to be ordered by "c":

```
drop table if exists t;
create table t(a bigint, b bigint, c bigint);
insert into t values(1, 2, 1), (1, 2, 2), (1, 3, 1), (1, 3, 2);
select distinct a, b from t order by c;
```

To order the result, duplicates must be eliminated first. But to do so, which row should we keep? This choice influences the retained value of "c", which in turn influences ordering and makes it arbitrary as well.

In MySQL, a query that has DISTINCT and ORDER BY is rejected as invalid if any ORDER
↪ BY expression does not satisfy at least one of these conditions:

- The expression is equal to one in the SELECT list
- All columns referenced by the expression and belonging to the query's selected tables are elements of the SELECT list

But in TiDB, the above query is legal, for more information see #4254.

Another TiDB extension to standard SQL permits references in the HAVING clause to aliased expressions in the SELECT list. For example, the following query returns "name" values that occur only once in table "orders":

```
select name, count(name) from orders
group by name
having count(name) = 1;
```

The TiDB extension permits the use of an alias in the HAVING clause for the aggregated column:

```
select name, count(name) as c from orders
group by name
having c = 1;
```

Standard SQL permits only column expressions in `GROUP BY` clauses, so a statement such as this is invalid because "FLOOR(value/100)" is a noncolumn expression:

```
select id, floor(value/100)
from tbl_name
group by id, floor(value/100);
```

TiDB extends standard SQL to permit noncolumn expressions in `GROUP BY` clauses and considers the preceding statement valid.

Standard SQL also does not permit aliases in `GROUP BY` clauses. TiDB extends standard SQL to permit aliases, so another way to write the query is as follows:

```
select id, floor(value/100) as val
from tbl_name
group by id, val;
```

4.1.4.13.4 Unsupported aggregate functions

The following aggregate functions are currently unsupported in TiDB. You can track our progress in TiDB #7623:

- `VARIANCE`, `VAR_POP`
- `JSON_ARRAYAGG`
- `JSON_OBJECTAGG`

4.1.4.14 Window Functions

The usage of window functions in TiDB is similar to that in MySQL 8.0. For details, see MySQL Window Functions.

Because window functions reserve additional words in the parser, TiDB provides an option to disable window functions. If you receive errors parsing SQL statements after upgrading, try setting `tidb_enable_window_function=0`.

TiDB supports the following window functions:

Function name	Feature description
`CUME_DIST()`	Returns the cumulative distribution of a value within a group of values.
`DENSE_RANK()`	Returns the rank of the current row within the partition, and the rank is without gaps.
`FIRST_VALUE()`	Returns the expression value of the first row in the current window.
`LAG()`	Returns the expression value from the row that precedes the current row by N rows within the partition.

Function name	Feature description
LAST_VALUE()	Returns the expression value of the last row in the current window.
LEAD()	Returns the expression value from the row that follows the current row by N rows within the partition.
NTH_VALUE()	Returns the expression value from the N-th row of the current window.
NTILE()	Divides a partition into N buckets, assigns the bucket number to each row in the partition, and returns the bucket number of the current row within the partition.
PERCENT_RANK()	Returns the percentage of partition values that are less than the value in the current row.
RANK()	Returns the rank of the current row within the partition. The rank may be with gaps.
ROW_NUMBER()	Returns the number of the current row in the partition.

4.1.4.15 Miscellaneous Functions

TiDB supports most of the miscellaneous functions available in MySQL 5.7.

4.1.4.15.1 Supported functions

Name	Description
ANY_VALUE()	Suppress ONLY_FULL_GROUP_BY value rejection
DEFAULT()	Returns the default value for a table column
INET_ATON()	Return the numeric value of an IP address
INET_NTOA()	Return the IP address from a numeric value
INET6_ATON()	Return the numeric value of an IPv6 address
INET6_NTOA()	Return the IPv6 address from a numeric value
IS_IPV4()	Whether argument is an IPv4 address
IS_IPV4_COMPAT()	Whether argument is an IPv4-compatible address
IS_IPV4_MAPPED()	Whether argument is an IPv4-mapped address
IS_IPV6()	Whether argument is an IPv6 address
NAME_CONST()	Can be used to rename a column name
SLEEP()	Sleep for a number of seconds
UUID()	Return a Universal Unique Identifier (UUID)
VALUES()	Defines the values to be used during an INSERT

4.1.4.15.2 Unsupported functions

Name	Description
GET_LOCK ()	Get a named lock TiDB #10929
RELEASE_LOCK ()	Releases the named lock TiDB #10929
UUID_SHORT ()	Provides a UUID that is unique given certain assumptions not present in TiDB TiDB #4620
MASTER_WAIT_POS ()	Relates to MySQL replication

4.1.4.16 Precision Math

The precision math support in TiDB is consistent with MySQL. For more information, see Precision Math in MySQL.

4.1.4.16.1 Numeric types

The scope of precision math for exact-value operations includes the exact-value data types (integer and DECIMAL types) and exact-value numeric literals. Approximate-value data types and numeric literals are handled as floating-point numbers.

Exact-value numeric literals have an integer part or fractional part, or both. They may be signed. Examples: `1`, `.2`, `3.4`, `-5`, `-6.78`, `+9.10`.

Approximate-value numeric literals are represented in scientific notation (power-of-10) with a mantissa and exponent. Either or both parts may be signed. Examples: `1.2E3`, `1.2E-3`, `-1.2E3`, `-1.2E-3`.

Two numbers that look similar might be treated differently. For example, `2.34` is an exact-value (fixed-point) number, whereas `2.34E0` is an approximate-value (floating-point) number.

The DECIMAL data type is a fixed-point type and the calculations are exact. The FLOAT and DOUBLE data types are floating-point types and calculations are approximate.

4.1.4.16.2 DECIMAL data type characteristics

This section discusses the following topics of the characteristics of the DECIMAL data type (and its synonyms):

- Maximum number of digits
- Storage format
- Storage requirements

The declaration syntax for a DECIMAL column is `DECIMAL(M,D)`. The ranges of values for the arguments are as follows:

- M is the maximum number of digits (the precision). 1<= M <= 65.
- D is the number of digits to the right of the decimal point (the scale). 1 <= D <= 30 and D must be no larger than M.

The maximum value of 65 for M means that calculations on DECIMAL values are accurate up to 65 digits. This limit of 65 digits of precision also applies to exact-value numeric literals.

Values for DECIMAL columns are stored using a binary format that packs 9 decimal digits into 4 bytes. The storage requirements for the integer and fractional parts of each value are determined separately. Each multiple of 9 digits requires 4 bytes, and any remaining digits left over require some fraction of 4 bytes. The storage required for remaining digits is given by the following table.

Leftover Digits	Number of Bytes
0	0
1–2	1
3–4	2
5–6	3
7–9	4

For example, a `DECIMAL(18,9)` column has 9 digits on each side of the decimal point, so the integer part and the fractional part each require 4 bytes. A `DECIMAL(20,6)` column has 14 integer digits and 6 fractional digits. The integer digits require 4 bytes for 9 of the digits and 3 bytes for the remaining 5 digits. The 6 fractional digits require 3 bytes.

DECIMAL columns do not store a leading `+` character or `-` character or leading `0` digits. If you insert `+0003.1` into a `DECIMAL(5,1)` column, it is stored as `3.1`. For negative numbers, a literal `-` character is not stored.

DECIMAL columns do not permit values larger than the range implied by the column definition. For example, a `DECIMAL(3,0)` column supports a range of `-999` to `999`. A `DECIMAL(M,D)` column permits at most `M - D` digits to the left of the decimal point.

For more information about the internal format of the DECIMAL values, see `mydecimal.go` in TiDB souce code.

4.1.4.16.3 Expression handling

For expressions with precision math, TiDB uses the exact-value numbers as given whenever possible. For example, numbers in comparisons are used exactly as given without a change in value. In strict SQL mode, if you add an exact data type into a column, a number is inserted with its exact value if it is within the column range. When retrieved, the value is

the same as what is inserted. If strict SQL mode is not enabled, truncation for INSERT is permitted in TiDB.

How to handle a numeric expression depends on the values of the expression:

- If the expression contains any approximate values, the result is approximate. TiDB evaluates the expression using floating-point arithmetic.
- If the expression contains no approximate values are present, which means only exact values are contained, and if any exact value contains a fractional part, the expression is evaluated using DECIMAL exact arithmetic and has a precision of 65 digits.
- Otherwise, the expression contains only integer values. The expression is exact. TiDB evaluates the expression using integer arithmetic and has a precision the same as BIGINT (64 bits).

If a numeric expression contains strings, the strings are converted to double-precision floating-point values and the result of the expression is approximate.

Inserts into numeric columns are affected by the SQL mode. The following discussions mention strict mode and `ERROR_FOR_DIVISION_BY_ZERO`. To turn on all the restrictions, you can simply use the `TRADITIONAL` mode, which includes both strict mode values and `ERROR_FOR_DIVISION_BY_ZERO`:

```
SET sql_mode = 'TRADITIONAL`;
```

If a number is inserted into an exact type column (DECIMAL or integer), it is inserted with its exact value if it is within the column range. For this number:

- If the value has too many digits in the fractional part, rounding occurs and a warning is generated.
- If the value has too many digits in the integer part, it is too large and is handled as follows:
  - If strict mode is not enabled, the value is truncated to the nearest legal value and a warning is generated.
  - If strict mode is enabled, an overflow error occurs.

To insert strings into numeric columns, TiDB handles the conversion from string to number as follows if the string has nonnumeric contents:

- In strict mode, a string (including an empty string) that does not begin with a number cannot be used as a number. An error, or a warning occurs.
- A string that begins with a number can be converted, but the trailing nonnumeric portion is truncated. In strict mode, if the truncated portion contains anything other than spaces, an error, or a warning occurs.

By default, the result of the division by 0 is NULL and no warning. By setting the SQL mode appropriately, division by 0 can be restricted. If you enable the `ERROR_FOR_DIVISION_BY_ZERO` SQL mode, TiDB handles division by 0 differently:

- In strict mode, inserts and updates are prohibited, and an error occurs.
- If it's not in the strict mode, a warning occurs.

In the following SQL statement:

```
INSERT INTO t SET i = 1/0;
```

The following results are returned in different SQL modes:

`sql_mode` Value	Result
''	No warning, no error; i is set to NULL.
strict	No warning, no error; i is set to NULL.
`ERROR_FOR_DIVISION_BY_ZERO`	Warning, no error; i is set to NULL.
strict, `ERROR_FOR_DIVISION_BY_ZERO`	Error; no row is inserted.

4.1.4.16.4 Rounding behavior

The result of the `ROUND()` function depends on whether its argument is exact or approximate:

- For exact-value numbers, the `ROUND()` function uses the "round half up" rule.
- For approximate-value numbers, the results in TiDB differs from that in MySQL:

```
TiDB > SELECT ROUND(2.5), ROUND(25E-1);
+------------+--------------+
| ROUND(2.5) | ROUND(25E-1) |
+------------+--------------+
|          3 |            3 |
+------------+--------------+
1 row in set (0.00 sec)
```

For inserts into a DECIMAL or integer column, the rounding uses round half away from zero.

```
TiDB > CREATE TABLE t (d DECIMAL(10,0));
Query OK, 0 rows affected (0.01 sec)

TiDB > INSERT INTO t VALUES(2.5),(2.5E0);
Query OK, 2 rows affected, 2 warnings (0.00 sec)
```

```
TiDB > SELECT d FROM t;
+------+
| d    |
+------+
|    3 |
|    3 |
+------+
2 rows in set (0.00 sec)
```

4.1.4.17 List of Expressions for Pushdown

When TiDB reads data from TiKV, TiDB tries to push down some expressions (including calculations of functions or operators) to be processed to TiKV. This reduces the amount of transferred data and offloads processing from a single TiDB node. This document introduces the expressions that TiDB already supports pushing down and how to prohibit specific expressions from being pushed down using blocklist.

4.1.4.17.1 Supported expressions for pushdown

Expression Type	Operations
Logical operators	AND (&&), OR (||), NOT (!)
Comparison functions and operators	<, <=, =, != (<>), >, >=, <=>, IN(), IS NULL, LIKE, IS TRUE, IS FALSE, COALESCE()
Numeric functions and operators	+, -, *, /, ABS(), CEIL(), CEILING(), FLOOR()
Control flow functions	CASE, IF(), IFNULL()
JSON functions	JSON_TYPE(json_val), JSON_EXTRACT(json_doc, path[, path] ...), JSON_UNQUOTE(json_val), JSON_OBJECT(key, val[, key, val] ...), JSON_ARRAY([val[, val] ...]), JSON_MERGE(json_doc, json_doc[, json_doc] ...), JSON_SET(json_doc, path, val[, path, val] ...), JSON_INSERT(json_doc, path, val[, path, val] ...), JSON_REPLACE(json_doc, path, val[, path, val] ...), JSON_REMOVE(json_doc, path[, path] ...)

Expression Type	Operations
Date and time functions	`DATE_FORMAT()`

4.1.4.17.2 Blocklist specific expressions

If unexpected behavior occurs during the calculation of a function caused by its push-down, you can quickly restore the application by blocklisting that function. Specifically, you can prohibit an expression from being pushed down by adding the corresponding functions or operator to the blocklist `mysql.expr_pushdown_blacklist`.

Add to the blocklist

To add one or more functions or operators to the blocklist, perform the following steps:

1. Insert the function or operator name to `mysql.expr_pushdown_blacklist`.
2. Execute the `admin reload expr_pushdown_blacklist;` command.

Remove from the blocklist

To remove one or more functions or operators from the blocklist, perform the following steps:

1. Delete the function or operator name in `mysql.expr_pushdown_blacklist`.
2. Execute the `admin reload expr_pushdown_blacklist;` command.

blocklist usage examples

The following example demonstrates how to add the `<` and `>` operators to the blocklist, then remove `>` from the blocklist.

You can see whether the blocklist takes effect by checking the results returned by `EXPLAIN` statement (See Understanding `EXPLAIN` results).

```
tidb> create table t(a int);
Query OK, 0 rows affected (0.01 sec)

tidb> explain select * from t where a < 2 and a > 2;
+--
    ↪ ------------------+----------+------+-----------------------------------------
    ↪
| id                | count    | task | operator info
    ↪                                      |
+--
    ↪ ------------------+----------+------+-----------------------------------------
    ↪
```

```
| TableReader_7    | 0.00    | root | data:Selection_6
   ↪                                  |
|  ─Selection_6    | 0.00    | cop | gt(test.t.a, 2), lt(test.t.a, 2)
   ↪                    |
|    ─TableScan_5   | 10000.00 | cop | table:t, range:[-inf,+inf], keep
   ↪ order:false, stats:pseudo |
+--
   ↪ ------------------+---------+------+-----------------------------------------
   ↪
3 rows in set (0.00 sec)

tidb> insert into mysql.expr_pushdown_blacklist values('<'), ('>');
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0

tidb> admin reload expr_pushdown_blacklist;
Query OK, 0 rows affected (0.00 sec)

tidb> explain select * from t where a < 2 and a > 2;
+--
   ↪ ------------------+---------+------+-----------------------------------------
   ↪
| id                | count   | task | operator info
   ↪                                       |
+--
   ↪ ------------------+---------+------+-----------------------------------------
   ↪
| Selection_5       | 8000.00 | root | gt(test.t.a, 2), lt(test.t.a, 2)
   ↪                    |
|  ─TableReader_7   | 10000.00 | root | data:TableScan_6
   ↪                                    |
|    ─TableScan_6   | 10000.00 | cop | table:t, range:[-inf,+inf], keep
   ↪ order:false, stats:pseudo |
+--
   ↪ ------------------+---------+------+-----------------------------------------
   ↪
3 rows in set (0.00 sec)

tidb> delete from mysql.expr_pushdown_blacklist where name = '>';
Query OK, 1 row affected (0.00 sec)

tidb> admin reload expr_pushdown_blacklist;
Query OK, 0 rows affected (0.00 sec)

tidb> explain select * from t where a < 2 and a > 2;
```

```
+--
   ↪ --------------------+---------+------+-----------------------------------
   ↪
| id                  | count   | task | operator info
   ↪                                      |
+--
   ↪ --------------------+---------+------+-----------------------------------
   ↪
| Selection_5         | 2666.67 | root | lt(test.t.a, 2)
   ↪                                     |
|  ─TableReader_8     | 3333.33 | root | data:Selection_7
   ↪                                    |
|    ─Selection_7     | 3333.33 | cop | gt(test.t.a, 2)
   ↪                                      |
|      ─TableScan_6   | 10000.00 | cop | table:t, range:[-inf,+inf], keep
   ↪ order:false, stats:pseudo |
+--
   ↪ --------------------+---------+------+-----------------------------------
   ↪
4 rows in set (0.00 sec)
```

Note:

- `admin reload expr_pushdown_blacklist` only takes effect on the TiDB server that executes this SQL statement. To make it apply to all TiDB servers, execute the SQL statement on each TiDB server.
- The feature of blocklisting specific expressions is supported in TiDB 3.0.0 or later versions.
- TiDB 3.0.3 or earlier versions does not support adding some of the operators (such as ">", "+", "is null") to the blocklist by using their original names. You need to use their aliases (case-sensitive) instead, as shown in the following table:

Operator Name	Aliases
<	lt
>	gt
<=	le
>=	ge
=	eq
!=	ne

Operator Name	Aliases
<>	ne
<=>	nulleq
|	bitor
&&	bitand
||	or
!	not
in	in
+	plus
-	minus
	mul
/	div
DIV	intdiv
IS NULL	isnull
IS TRUE	istrue
IS FALSE	isfalse

4.1.5 SQL Statements

4.1.5.1 ADD COLUMN

The `ALTER TABLE.. ADD COLUMN` statement adds a column to an existing table. This operation is online in TiDB, which means that neither reads or writes to the table are blocked by adding a column.

4.1.5.1.1 Synopsis

```
AlterTableStmt ::=
    'ALTER' IgnoreOptional 'TABLE' TableName ( AlterTableSpecListOpt
       ↪ AlterTablePartitionOpt | 'ANALYZE' 'PARTITION' PartitionNameList (
       ↪  'INDEX' IndexNameList )? AnalyzeOptionListOpt )

AlterTableSpec ::=
    TableOptionList
|   'SET' 'TIFLASH' 'REPLICA' LengthNum LocationLabelList
|   'CONVERT' 'TO' CharsetKw ( CharsetName | 'DEFAULT' ) OptCollate
|   'ADD' ( ColumnKeywordOpt IfNotExists ( ColumnDef ColumnPosition | '('
    ↪ TableElementList ')' ) | Constraint | 'PARTITION' IfNotExists
    ↪ NoWriteToBinLogAliasOpt ( PartitionDefinitionListOpt | 'PARTITIONS'
    ↪ NUM ) )
|   ( ( 'CHECK' | 'TRUNCATE' ) 'PARTITION' | ( 'OPTIMIZE' | 'REPAIR' | '
    ↪ REBUILD' ) 'PARTITION' NoWriteToBinLogAliasOpt )
    ↪ AllOrPartitionNameList
|   'COALESCE' 'PARTITION' NoWriteToBinLogAliasOpt NUM
```

```
|   'DROP' ( ColumnKeywordOpt IfExists ColumnName RestrictOrCascadeOpt | '
    ↪ PRIMARY' 'KEY' | 'PARTITION' IfExists PartitionNameList | (
    ↪ KeyOrIndex IfExists | 'CHECK' ) Identifier | 'FOREIGN' 'KEY' IfExists
    ↪  Symbol )
|   'EXCHANGE' 'PARTITION' Identifier 'WITH' 'TABLE' TableName
    ↪ WithValidationOpt
|   ( 'IMPORT' | 'DISCARD' ) ( 'PARTITION' AllOrPartitionNameList )? '
    ↪ TABLESPACE'
|   'REORGANIZE' 'PARTITION' NoWriteToBinLogAliasOpt
    ↪ ReorganizePartitionRuleOpt
|   'ORDER' 'BY' AlterOrderItem ( ',' AlterOrderItem )*
|   ( 'DISABLE' | 'ENABLE' ) 'KEYS'
|   ( 'MODIFY' ColumnKeywordOpt IfExists | 'CHANGE' ColumnKeywordOpt
    ↪ IfExists ColumnName ) ColumnDef ColumnPosition
|   'ALTER' ( ColumnKeywordOpt ColumnName ( 'SET' 'DEFAULT' ( SignedLiteral
    ↪ | '(' Expression ')' ) | 'DROP' 'DEFAULT' ) | 'CHECK' Identifier
    ↪ EnforcedOrNot | 'INDEX' Identifier IndexInvisible )
|   'RENAME' ( ( 'COLUMN' | KeyOrIndex ) Identifier 'TO' Identifier | ( 'TO'
    ↪  | '='? | 'AS' ) TableName )
|   LockClause
|   AlgorithmClause
|   'FORCE'
|   ( 'WITH' | 'WITHOUT' ) 'VALIDATION'
|   'SECONDARY_LOAD'
|   'SECONDARY_UNLOAD'

ColumnKeywordOpt ::=
    'COLUMN'?

ColumnDef ::=
    ColumnName ( Type | 'SERIAL' ) ColumnOptionListOpt

ColumnPosition ::=
    ( 'FIRST' | 'AFTER' ColumnName )?
```

4.1.5.1.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 VALUES (NULL);
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM t1;
```

```
+----+
| id |
+----+
|  1 |
+----+
1 row in set (0.00 sec)

mysql> ALTER TABLE t1 ADD COLUMN c1 INT NOT NULL;
Query OK, 0 rows affected (0.28 sec)

mysql> SELECT * FROM t1;
+----+----+
| id | c1 |
+----+----+
|  1 |  0 |
+----+----+
1 row in set (0.00 sec)

mysql> ALTER TABLE t1 ADD c2 INT NOT NULL AFTER c1;
Query OK, 0 rows affected (0.28 sec)

mysql> SELECT * FROM t1;
+----+----+----+
| id | c1 | c2 |
+----+----+----+
|  1 |  0 |  0 |
+----+----+----+
1 row in set (0.00 sec)
```

4.1.5.1.3 MySQL compatibility

- Adding multiple columns at the same time is currently not supported.
- Adding a new column and setting it to the `PRIMARY KEY` is not supported.
- Adding a new column and setting it to `AUTO_INCREMENT` is not supported.

4.1.5.1.4 See also

- ADD INDEX
- CREATE TABLE

4.1.5.2 ADD INDEX

The `ALTER TABLE.. ADD INDEX` statement adds an index to an existing table. This operation is online in TiDB, which means that neither reads or writes to the table are blocked by adding an index.

4.1.5.2.1 Synopsis

```
AlterTableStmt ::=
    'ALTER' IgnoreOptional 'TABLE' TableName ( AlterTableSpecListOpt
       ↪ AlterTablePartitionOpt | 'ANALYZE' 'PARTITION' PartitionNameList (
       ↪  'INDEX' IndexNameList )? AnalyzeOptionListOpt )

AlterTableSpec ::=
    TableOptionList
|   'SET' 'TIFLASH' 'REPLICA' LengthNum LocationLabelList
|   'CONVERT' 'TO' CharsetKw ( CharsetName | 'DEFAULT' ) OptCollate
|   'ADD' ( ColumnKeywordOpt IfNotExists ( ColumnDef ColumnPosition | '('
    ↪ TableElementList ')' ) | Constraint | 'PARTITION' IfNotExists
    ↪ NoWriteToBinLogAliasOpt ( PartitionDefinitionListOpt | 'PARTITIONS'
    ↪ NUM ) )
|   ( ( 'CHECK' | 'TRUNCATE' ) 'PARTITION' | ( 'OPTIMIZE' | 'REPAIR' | '
    ↪ REBUILD' ) 'PARTITION' NoWriteToBinLogAliasOpt )
    ↪ AllOrPartitionNameList
|   'COALESCE' 'PARTITION' NoWriteToBinLogAliasOpt NUM
|   'DROP' ( ColumnKeywordOpt IfExists ColumnName RestrictOrCascadeOpt | '
    ↪ PRIMARY' 'KEY' | 'PARTITION' IfExists PartitionNameList | (
    ↪ KeyOrIndex IfExists | 'CHECK' ) Identifier | 'FOREIGN' 'KEY' IfExists
    ↪  Symbol )
|   'EXCHANGE' 'PARTITION' Identifier 'WITH' 'TABLE' TableName
    ↪ WithValidationOpt
|   ( 'IMPORT' | 'DISCARD' ) ( 'PARTITION' AllOrPartitionNameList )? '
    ↪ TABLESPACE'
|   'REORGANIZE' 'PARTITION' NoWriteToBinLogAliasOpt
    ↪ ReorganizePartitionRuleOpt
|   'ORDER' 'BY' AlterOrderItem ( ',' AlterOrderItem )*
|   ( 'DISABLE' | 'ENABLE' ) 'KEYS'
|   ( 'MODIFY' ColumnKeywordOpt IfExists | 'CHANGE' ColumnKeywordOpt
    ↪ IfExists ColumnName ) ColumnDef ColumnPosition
|   'ALTER' ( ColumnKeywordOpt ColumnName ( 'SET' 'DEFAULT' ( SignedLiteral
    ↪ | '(' Expression ')' ) | 'DROP' 'DEFAULT' ) | 'CHECK' Identifier
    ↪ EnforcedOrNot | 'INDEX' Identifier IndexInvisible )
|   'RENAME' ( ( 'COLUMN' | KeyOrIndex ) Identifier 'TO' Identifier | ( 'TO'
    ↪  | '='? | 'AS' ) TableName )
|   LockClause
|   AlgorithmClause
|   'FORCE'
```

```
|   ( 'WITH' | 'WITHOUT' ) 'VALIDATION'
|   'SECONDARY_LOAD'
|   'SECONDARY_UNLOAD'

ColumnKeywordOpt ::=
    'COLUMN'?

ColumnDef ::=
    ColumnName ( Type | 'SERIAL' ) ColumnOptionListOpt

ColumnPosition ::=
    ( 'FIRST' | 'AFTER' ColumnName )?
```

4.1.5.2.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 (c1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.03 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> EXPLAIN SELECT * FROM t1 WHERE c1 = 3;
+--
    ↪ ------------------+---------+------+-----------------------------------
    ↪
| id                | count   | task | operator info
    ↪                                      |
+--
    ↪ ------------------+---------+------+-----------------------------------
    ↪
| TableReader_7     | 10.00   | root | data:Selection_6
    ↪                                       |
|  ─Selection_6     | 10.00   | cop | eq(test.t1.c1, 3)
    ↪                                       |
|    ─TableScan_5   | 10000.00 | cop | table:t1, range:[-inf,+inf], keep
    ↪ order:false, stats:pseudo |
+--
    ↪ ------------------+---------+------+-----------------------------------
    ↪
3 rows in set (0.00 sec)

mysql> ALTER TABLE t1 ADD INDEX (c1);
```

```
Query OK, 0 rows affected (0.30 sec)

mysql> EXPLAIN SELECT * FROM t1 WHERE c1 = 3;
+--
    ↪ ----------------+-------+------+-----------------------------------------------
    ↪
| id              | count | task | operator info
    ↪                                          |
+--
    ↪ ----------------+-------+------+-----------------------------------------------
    ↪
| IndexReader_6   | 10.00 | root | index:IndexScan_5
    ↪                                        |
| └─IndexScan_5   | 10.00 | cop | table:t1, index:c1, range:[3,3], keep
    ↪ order:false, stats:pseudo |
+--
    ↪ ----------------+-------+------+-----------------------------------------------
    ↪
2 rows in set (0.00 sec)
```

4.1.5.2.3 MySQL compatibility

- `FULLTEXT`, `HASH` and `SPATIAL` indexes are not supported.
- Descending indexes are not supported (similar to MySQL 5.7).
- Adding multiple indexes at the same time is currently not supported.
- Adding the primary key constraint to a table is not supported by default. You can enable the feature by setting the `alter-primary-key` configuration item to `true`. For details, see alter-primary-key.

4.1.5.2.4 See also

- CREATE INDEX
- DROP INDEX
- RENAME INDEX
- ADD COLUMN
- CREATE TABLE
- EXPLAIN

4.1.5.3 ADMIN

This statement is a TiDB extension syntax, used to view the status of TiDB and check the data of tables in TiDB.

To view the currently running DDL jobs, use `ADMIN SHOW DDL`:

```
ADMIN SHOW DDL;
```

To view all the results in the current DDL job queue (including tasks that are running and waiting to be run) and the last ten results in the completed DDL job queue, use `ADMIN SHOW DDL JOBS`:

```
ADMIN SHOW DDL JOBS;
```

To view the original SQL statements of the DDL job corresponding to `job_id`, use `ADMIN SHOW DDL JOB QUERIES`:

```
ADMIN SHOW DDL JOB QUERIES job_id [, job_id] ...;
```

You can only searches the running DDL job corresponding to `job_id` and the last ten results in the DDL history job queue.

To cancel the currently running DDL jobs and return whether the corresponding jobs are successfully cancelled, use `ADMIN CANCEL DDL JOBS`:

```
ADMIN CANCEL DDL JOBS job_id [, job_id] ...;
```

If the operation fails to cancel the jobs, specific reasons are displayed.

Note:

- Only this operation can cancel DDL jobs. All other operations and environment changes (such as machine restart and cluster restart) cannot cancel these jobs.
- This operation can cancel multiple DDL jobs at the same time. You can get the ID of DDL jobs using the `ADMIN SHOW DDL JOBS` statement.
- If the jobs you want to cancel are finished, the cancellation operation fails.

To check the consistency of all the data and corresponding indexes in the `tbl_name` table, use `ADMIN CHECK TABLE:`

```
ADMIN CHECK TABLE tbl_name [, tbl_name] ...;
```

If the consistency check is passed, an empty result is returned. Otherwise, an error message is returned indicating that the data is inconsistent.

4.1.5.3.1 Synopsis

```
AdminStmt ::=
    'ADMIN' ( 'SHOW' ( 'DDL' ( 'JOBS' Int64Num? WhereClauseOptional | 'JOB'
       ↪ 'QUERIES' NumList )? | TableName 'NEXT_ROW_ID' | 'SLOW'
       ↪ AdminShowSlow ) | 'CHECK' ( 'TABLE' TableNameList | 'INDEX'
       ↪ TableName Identifier ( HandleRange ( ',' HandleRange )* )? ) | '
       ↪ RECOVER' 'INDEX' TableName Identifier | 'CLEANUP' ( 'INDEX'
       ↪ TableName Identifier | 'TABLE' 'LOCK' TableNameList ) | 'CHECKSUM'
       ↪  'TABLE' TableNameList | 'CANCEL' 'DDL' 'JOBS' NumList | 'RELOAD'
       ↪ ( 'EXPR_PUSHDOWN_BLACKLIST' | 'OPT_RULE_BLACKLIST' | 'BINDINGS' )
       ↪ | 'PLUGINS' ( 'ENABLE' | 'DISABLE' ) PluginNameList | 'REPAIR' '
       ↪ TABLE' TableName CreateTableStmt | ( 'FLUSH' | 'CAPTURE' | 'EVOLVE
       ↪ ' ) 'BINDINGS' )
```

4.1.5.3.2 Examples

```
mysql> admin show ddl jobs;
+--
   ↪ ------+---------+------------+--------------+----------------------+----------
   ↪
| JOB_ID | DB_NAME | TABLE_NAME | JOB_TYPE | SCHEMA_STATE     | SCHEMA_ID |
   ↪  TABLE_ID | ROW_COUNT | START_TIME                | STATE       |
+--
   ↪ ------+---------+------------+--------------+----------------------+----------
   ↪
| 45     | test    | t1         | add index    | write reorganization | 32    |
   ↪ 37      | 0         | 2019-01-10 12:38:36.501 +0800 CST | running |
| 44     | test    | t1         | add index    | none                 | 32    |
   ↪ 37      | 0         | 2019-01-10 12:36:55.18 +0800 CST | rollback done
   ↪ |
| 43     | test    | t1         | add index    | public               | 32    |
   ↪ 37      | 6         | 2019-01-10 12:35:13.66 +0800 CST | synced   |
| 42     | test    | t1         | drop index   | none                 | 32    |
   ↪ 37      | 0         | 2019-01-10 12:34:35.204 +0800 CST | synced |
| 41     | test    | t1         | add index    | public               | 32    |
   ↪ 37      | 0         | 2019-01-10 12:33:22.62 +0800 CST | synced   |
| 40     | test    | t1         | drop column  | none                 | 32    |
   ↪ 37      | 0         | 2019-01-10 12:33:08.212 +0800 CST | synced |
| 39     | test    | t1         | add column   | public               | 32    |
   ↪ 37      | 0         | 2019-01-10 12:32:55.42 +0800 CST | synced   |
| 38     | test    | t1         | create table | public               | 32    |
   ↪ 37      | 0         | 2019-01-10 12:32:41.956 +0800 CST | synced |
| 36     | test    |            | drop table   | none                 | 32    |
   ↪ 34      | 0         | 2019-01-10 11:29:59.982 +0800 CST | synced |
```

```
| 35    | test   |           | create table | public           | 32       |
   ↪ 34      | 0        | 2019-01-10 11:29:40.741 +0800 CST | synced  |
| 33    | test   |           | create schema | public           | 32       | 0
   ↪         | 0        | 2019-01-10 11:29:22.813 +0800 CST | synced  |
+--
   ↪ ------+---------+-----------+--------------+---------------------+----------
   ↪
```

- `JOB_ID`: each DDL operation corresponds to one DDL job. `JOB_ID` is globally unique.
- `DB_NAME`: the name of the database on which the DDL operations are performed.
- `TABLE_NAME`: the name of the table on which the DDL operations are performed.
- `JOB_TYPE`: the type of the DDL operations.
- `SCHEMA_STATE`: the current state of the schema. If the `JOB_TYPE` is `add index`, it is the state of the index; if the `JOB_TYPE` is `add column`, it is the state of the column; if the `JOB_TYPE` is `create table`, it is the state of the table. The common states include:
  - `none`: it indicates not existing. When the `drop` or `create` operation fails and rolls back, it usually becomes the `none` state.
  - `delete only`, `write only`, `delete reorganization`, `write reorganization`: these four states are intermediate states. For details, see the paper Online, Asynchronous Schema Change in F1. These states are not visible in common operations, because the conversion from the intermediate states is so quick. You can see the `write reorganization` state only in `add index` operations, which means that the index data is being added.
  - `public`: it indicates existing and usable. When operations like `create table` and `add index/column` are finished, it usually becomes the `public` state, which means that the created table/column/index can be normally read and written now.
- `SCHEMA_ID`: the ID of the database on which the DDL operations are performed.
- `TABLE_ID`: the ID of the table on which the DDL operations are performed.
- `ROW_COUNT`: the number of the data rows that have been added when running the `add index` operation.
- `START_TIME`: the start time of the DDL operations.
- `STATE`: the state of the DDL operations. The common states include:
  - `none`: it indicates that the operation task has been put in the DDL job queue but has not been performed yet, because it is waiting for the previous tasks to complete. Another reason might be that it becomes the `none` state after running the drop operation, but it will soon be updated to the `synced` state, which means that all TiDB instances have been synced to this state.
  - `running`: it indicates that the operation is being performed.
  - `synced`: it indicates that the operation has been performed successfully and all TiDB instances have been synced to this state.
  - `rollback done`: it indicates that the operation has failed and has finished rolling back.

- `rollingback`: it indicates that the operation has failed and is rolling back.
- `cancelling`: it indicates that the operation is being cancelled. This state only occurs when you cancel DDL jobs using the `ADMIN CANCEL DDL JOBS` command.

4.1.5.3.3 MySQL compatibility

This statement is a TiDB extension to MySQL syntax.

4.1.5.4 ADMIN CANCEL DDL

The `ADMIN CANCEL DDL` statement allows you to cancel a running DDL job. The `job_id` can be found by running `ADMIN SHOW DDL JOBS`.

4.1.5.4.1 Synopsis

```
AdminStmt ::=
    'ADMIN' ( 'SHOW' ( 'DDL' ( 'JOBS' Int64Num? WhereClauseOptional | 'JOB'
       ↪ 'QUERIES' NumList )? | TableName 'NEXT_ROW_ID' | 'SLOW'
       ↪ AdminShowSlow ) | 'CHECK' ( 'TABLE' TableNameList | 'INDEX'
       ↪ TableName Identifier ( HandleRange ( ',' HandleRange )* )? ) | '
       ↪ RECOVER' 'INDEX' TableName Identifier | 'CLEANUP' ( 'INDEX'
       ↪ TableName Identifier | 'TABLE' 'LOCK' TableNameList ) | 'CHECKSUM'
       ↪  'TABLE' TableNameList | 'CANCEL' 'DDL' 'JOBS' NumList | 'RELOAD'
       ↪ ( 'EXPR_PUSHDOWN_BLACKLIST' | 'OPT_RULE_BLACKLIST' | 'BINDINGS' )
       ↪ | 'PLUGINS' ( 'ENABLE' | 'DISABLE' ) PluginNameList | 'REPAIR' '
       ↪ TABLE' TableName CreateTableStmt | ( 'FLUSH' | 'CAPTURE' | 'EVOLVE
       ↪ ' ) 'BINDINGS' )

NumList ::=
    Int64Num ( ',' Int64Num )*
```

4.1.5.4.2 Examples

To cancel the currently running DDL jobs and return whether the corresponding jobs are successfully cancelled, use `ADMIN CANCEL DDL JOBS`:

```
ADMIN CANCEL DDL JOBS job_id [, job_id] ...;
```

If the operation fails to cancel the jobs, specific reasons are displayed.

Note:

- Only this operation can cancel DDL jobs. All other operations and environment changes (such as machine restart and cluster restart) cannot cancel these jobs.
- This operation can cancel multiple DDL jobs at the same time. You can get the ID of DDL jobs using the `ADMIN SHOW DDL JOBS` statement.
- If the jobs you want to cancel are finished, the cancellation operation fails.

4.1.5.4.3 MySQL compatibility

This statement is a TiDB extension to MySQL syntax.

4.1.5.4.4 See also

- `ADMIN SHOW DDL [JOBS|QUERIES]`

4.1.5.5 ADMIN CHECKSUM TABLE

The `ADMIN CHECKSUM TABLE` statement calculates a CRC64 checksum for the data and indexes of a table. This statement is used by programs such as TiDB Lightning to ensure that import operations have completed successfully.

4.1.5.5.1 Synopsis

```
AdminStmt ::=
    'ADMIN' ( 'SHOW' ( 'DDL' ( 'JOBS' Int64Num? WhereClauseOptional | 'JOB'
       ↪ 'QUERIES' NumList )? | TableName 'NEXT_ROW_ID' | 'SLOW'
       ↪ AdminShowSlow ) | 'CHECK' ( 'TABLE' TableNameList | 'INDEX'
       ↪ TableName Identifier ( HandleRange ( ',' HandleRange )* )? ) | '
       ↪ RECOVER' 'INDEX' TableName Identifier | 'CLEANUP' ( 'INDEX'
       ↪ TableName Identifier | 'TABLE' 'LOCK' TableNameList ) | 'CHECKSUM'
       ↪  'TABLE' TableNameList | 'CANCEL' 'DDL' 'JOBS' NumList | 'RELOAD'
       ↪ ( 'EXPR_PUSHDOWN_BLACKLIST' | 'OPT_RULE_BLACKLIST' | 'BINDINGS' )
       ↪ | 'PLUGINS' ( 'ENABLE' | 'DISABLE' ) PluginNameList | 'REPAIR' '
       ↪ TABLE' TableName CreateTableStmt | ( 'FLUSH' | 'CAPTURE' | 'EVOLVE
       ↪ ' ) 'BINDINGS' )

TableNameList ::=
    TableName ( ',' TableName )*
```

4.1.5.5.2 Examples

Calculate the checksum for a table:

```
CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY auto_increment);
INSERT INTO t1 VALUES (1),(2),(3);
ADMIN CHECKSUM TABLE t1;
```

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY auto_increment);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 VALUES (1),(2),(3);
Query OK, 3 rows affected (0.02 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> ADMIN CHECKSUM TABLE t1;
+---------+------------+---------------------+-----------+-------------+
| Db_name | Table_name | Checksum_crc64_xor | Total_kvs | Total_bytes |
+---------+------------+---------------------+-----------+-------------+
| test    | t1         | 10909174369497628533 |     3 |         75 |
+---------+------------+---------------------+-----------+-------------+
1 row in set (0.00 sec)
```

4.1.5.5.3 MySQL compatibility

This statement is a TiDB extension to MySQL syntax.

4.1.5.6 ADMIN CHECK [TABLE|INDEX]

The `ADMIN CHECK [TABLE|INDEX]` statement checks for data consistency of tables and indexes.

4.1.5.6.1 Synopsis

```
AdminStmt ::=
    'ADMIN' ( 'SHOW' ( 'DDL' ( 'JOBS' Int64Num? WhereClauseOptional | 'JOB'
       ↪ 'QUERIES' NumList )? | TableName 'NEXT_ROW_ID' | 'SLOW'
       ↪ AdminShowSlow ) | 'CHECK' ( 'TABLE' TableNameList | 'INDEX'
       ↪ TableName Identifier ( HandleRange ( ',' HandleRange )* )? ) | '
       ↪ RECOVER' 'INDEX' TableName Identifier | 'CLEANUP' ( 'INDEX'
       ↪ TableName Identifier | 'TABLE' 'LOCK' TableNameList ) | 'CHECKSUM'
       ↪  'TABLE' TableNameList | 'CANCEL' 'DDL' 'JOBS' NumList | 'RELOAD'
       ↪ ( 'EXPR_PUSHDOWN_BLACKLIST' | 'OPT_RULE_BLACKLIST' | 'BINDINGS' )
       ↪ | 'PLUGINS' ( 'ENABLE' | 'DISABLE' ) PluginNameList | 'REPAIR' '
       ↪ TABLE' TableName CreateTableStmt | ( 'FLUSH' | 'CAPTURE' | 'EVOLVE
       ↪ ' ) 'BINDINGS' )
```

```
TableNameList ::=
    TableName ( ',' TableName )*
```

4.1.5.6.2 Examples

To check the consistency of all the data and corresponding indexes in the `tbl_name` table, use `ADMIN CHECK TABLE`:

```
ADMIN CHECK TABLE tbl_name [, tbl_name] ...;
```

If the consistency check is passed, an empty result is returned. Otherwise, an error message is returned indicating that the data is inconsistent.

```
ADMIN CHECK INDEX tbl_name idx_name;
```

The above statement is used to check the consistency of the column data and index data corresponding to the `idx_name` index in the `tbl_name` table. If the consistency check is passed, an empty result is returned; otherwise, an error message is returned indicating that the data is inconsistent.

```
ADMIN CHECK INDEX tbl_name idx_name (lower_val, upper_val) [, (lower_val,
    ↪ upper_val)] ...;
```

The above statement is used to check the consistency of the column data and index data corresponding to the `idx_name` index in the `tbl_name` table, with the data range (to be checked) specified. If the consistency check is passed, an empty result is returned. Otherwise, an error message is returned indicating that the data is inconsistent.

4.1.5.6.3 MySQL compatibility

This statement is a TiDB extension to MySQL syntax.

4.1.5.6.4 See also

- ADMIN

4.1.5.7 ADMIN SHOW DDL [JOBS|QUERIES]

The `ADMIN SHOW DDL [JOBS|QUERIES]` statement shows information about running and recently completed DDL jobs.

4.1.5.7.1 Synopsis

```
AdminStmt ::=
    'ADMIN' ( 'SHOW' ( 'DDL' ( 'JOBS' Int64Num? WhereClauseOptional | 'JOB'
       ↪ 'QUERIES' NumList )? | TableName 'NEXT_ROW_ID' | 'SLOW'
       ↪ AdminShowSlow ) | 'CHECK' ( 'TABLE' TableNameList | 'INDEX'
       ↪ TableName Identifier ( HandleRange ( ',' HandleRange )* )? ) | '
       ↪ RECOVER' 'INDEX' TableName Identifier | 'CLEANUP' ( 'INDEX'
       ↪ TableName Identifier | 'TABLE' 'LOCK' TableNameList ) | 'CHECKSUM'
       ↪  'TABLE' TableNameList | 'CANCEL' 'DDL' 'JOBS' NumList | 'RELOAD'
       ↪ ( 'EXPR_PUSHDOWN_BLACKLIST' | 'OPT_RULE_BLACKLIST' | 'BINDINGS' )
       ↪ | 'PLUGINS' ( 'ENABLE' | 'DISABLE' ) PluginNameList | 'REPAIR' '
       ↪ TABLE' TableName CreateTableStmt | ( 'FLUSH' | 'CAPTURE' | 'EVOLVE
       ↪ ' ) 'BINDINGS' )

NumList ::=
    Int64Num ( ',' Int64Num )*

WhereClauseOptional ::=
    WhereClause?
```

4.1.5.7.2 Examples

`ADMIN SHOW DDL`

To view the currently running DDL jobs, use `ADMIN SHOW DDL`:

```
ADMIN SHOW DDL;
```

```
mysql> ADMIN SHOW DDL;
+--
   ↪ ----------+--------------------------------------+--------------+-------------
   ↪
| SCHEMA_VER | OWNER_ID                             | OWNER_ADDRESS | RUNNING_JOBS
   ↪  | SELF_ID                      | QUERY |
+--
   ↪ ----------+--------------------------------------+--------------+-------------
   ↪
|         26 | 2d1982af-fa63-43ad-a3d5-73710683cc63 | 0.0.0.0:4000 |      | 2
   ↪ d1982af-fa63-43ad-a3d5-73710683cc63 | |
+--
   ↪ ----------+--------------------------------------+--------------+-------------
   ↪
1 row in set (0.00 sec)
```

`ADMIN SHOW DDL JOBS`

To view all the results in the current DDL job queue (including tasks that are running and waiting to be run) and the last ten results in the completed DDL job queue, use `ADMIN SHOW DDL JOBS`:

```
ADMIN SHOW DDL JOBS;
```

```
mysql> ADMIN SHOW DDL JOBS;
+--
   ↪ ------+---------+------------------+-------------+---------------------+----
   ↪
| JOB_ID | DB_NAME | TABLE_NAME     | JOB_TYPE    | SCHEMA_STATE       |
   ↪ SCHEMA_ID | TABLE_ID | ROW_COUNT | START_TIME    | END_TIME           |
   ↪ STATE  |
+--
   ↪ ------+---------+------------------+-------------+---------------------+----
   ↪
|     59 | test   | t1               | add index   | write reorganization |
   ↪      1 |       55 |    88576 | 2020-08-17 07:51:58 | NULL           |
   ↪ running |
|     60 | test   | t2               | add index   | none                 |
   ↪         1 |       57 |        0 | 2020-08-17 07:51:59 | NULL           |
   ↪ none   |
|     58 | test   | t2               | create table | public              |
   ↪         1 |       57 |        0 | 2020-08-17 07:41:28 | 2020-08-17
   ↪ 07:41:28 | synced |
|     56 | test   | t1               | create table | public              |
   ↪         1 |       55 |        0 | 2020-08-17 07:41:02 | 2020-08-17
   ↪ 07:41:02 | synced |
|     54 | test   | t1               | drop table  | none                 |
   ↪         1 |       50 |        0 | 2020-08-17 07:41:02 | 2020-08-17
   ↪ 07:41:02 | synced |
|     53 | test   | t1               | drop index  | none                 |
   ↪         1 |       50 |        0 | 2020-08-17 07:35:44 | 2020-08-17
   ↪ 07:35:44 | synced |
|     52 | test   | t1               | add index   | public               |
   ↪         1 |       50 |   451010 | 2020-08-17 07:34:43 | 2020-08-17
   ↪ 07:35:16 | synced |
|     51 | test   | t1               | create table | public              |
   ↪         1 |       50 |        0 | 2020-08-17 07:34:02 | 2020-08-17
   ↪ 07:34:02 | synced |
|     49 | test   | t1               | drop table  | none                 |
   ↪         1 |       47 |        0 | 2020-08-17 07:34:02 | 2020-08-17
   ↪ 07:34:02 | synced |
|     48 | test   | t1               | create table | public              |
   ↪         1 |       47 |        0 | 2020-08-17 07:33:37 | 2020-08-17
```

```
    ↪ 07:33:37 | synced |
|    46 | mysql  | stats_extended   | create table | public          |
    ↪        3 |      45 |        0 | 2020-08-17 06:42:38 | 2020-08-17
    ↪ 06:42:38 | synced |
|    44 | mysql  | opt_rule_blacklist | create table | public        |
    ↪        3 |      43 |        0 | 2020-08-17 06:42:38 | 2020-08-17
    ↪ 06:42:38 | synced |
+--
    ↪ ------+---------+--------------------+-------------+---------------------+----
    ↪
12 rows in set (0.00 sec)
```

From the output above:

- Job 59 is currently in progress (`STATE` of `running`). The schema state is currently in `write reorganization`, but will switch to `public` once the task is completed to note that the change can be observed publicly by user sessions. The `end_time` column is also `NULL` indicating that the completion time for the job is currently not known.
- Job 60 is an `add index` job, which is currently queued waiting for job 59 to complete. When job 59 completes, the `STATE` of job 60 will switch to `running`.
- For destructive changes such as dropping an index or dropping a table, the `SCHEMA_STATE` will change to `none` when the job is complete. For additive changes, the `SCHEMA_STATE` will change to `public`.

To limit the number of rows shown, specify a number and a where condition:

```
ADMIN SHOW DDL JOBS [NUM] [WHERE where_condition];
```

- `NUM`: to view the last `NUM` results in the completed DDL job queue. If not specified, `NUM` is by default 10.
- `WHERE`: to add filter conditions.

`ADMIN SHOW DDL JOB QUERIES`

To view the original SQL statements of the DDL job corresponding to `job_id`, use `ADMIN SHOW DDL JOB QUERIES`:

```
ADMIN SHOW DDL JOBS;
ADMIN SHOW DDL JOB QUERIES 51;
```

```
mysql> ADMIN SHOW DDL JOB QUERIES 51;
+--------------------------------------------------------------+
| QUERY                                                        |
```

```
+-------------------------------------------------------------+
| CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY auto_increment) |
+-------------------------------------------------------------+
1 row in set (0.02 sec)
```

You can only search the running DDL job corresponding to `job_id` within the last ten results in the DDL history job queue.

4.1.5.7.3 MySQL compatibility

This statement is a TiDB extension to MySQL syntax.

4.1.5.7.4 See also

- ADMIN CANCEL DDL

4.1.5.8 ALTER DATABASE

`ALTER DATABASE` is used to specify or modify the default character set and collation of the current database. `ALTER SCHEMA` has the same effect as `ALTER DATABASE`.

4.1.5.8.1 Examples

```
ALTER {DATABASE | SCHEMA} [db_name]
    alter_specification ...
alter_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
```

The `alter_specification` option specifies the `CHARACTER SET` and `COLLATE` of a specified database. Currently, TiDB only supports some character sets and collations. See Character Set Support for details.

4.1.5.8.2 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.8.3 See also

- CREATE DATABASE
- SHOW DATABASES

4.1.5.9 ALTER TABLE

This statement modifies an existing table to conform to a new table structure. The statement `ALTER TABLE` can be used to:

- `ADD`, `DROP`, or `RENAME` indexes
- `ADD`, `DROP`, `MODIFY` or `CHANGE` columns

4.1.5.9.1 Synopsis

```
AlterTableStmt ::=
    'ALTER' IgnoreOptional 'TABLE' TableName ( AlterTableSpecListOpt
       ↪ AlterTablePartitionOpt | 'ANALYZE' 'PARTITION' PartitionNameList (
       ↪  'INDEX' IndexNameList )? AnalyzeOptionListOpt )

TableName ::=
    Identifier ('.' Identifier)?

AlterTableSpec ::=
    TableOptionList
|   'SET' 'TIFLASH' 'REPLICA' LengthNum LocationLabelList
|   'CONVERT' 'TO' CharsetKw ( CharsetName | 'DEFAULT' ) OptCollate
|   'ADD' ( ColumnKeywordOpt IfNotExists ( ColumnDef ColumnPosition | '('
    ↪ TableElementList ')' ) | Constraint | 'PARTITION' IfNotExists
    ↪ NoWriteToBinLogAliasOpt ( PartitionDefinitionListOpt | 'PARTITIONS'
    ↪ NUM ) )
|   ( ( 'CHECK' | 'TRUNCATE' ) 'PARTITION' | ( 'OPTIMIZE' | 'REPAIR' | '
    ↪ REBUILD' ) 'PARTITION' NoWriteToBinLogAliasOpt )
    ↪ AllOrPartitionNameList
|   'COALESCE' 'PARTITION' NoWriteToBinLogAliasOpt NUM
|   'DROP' ( ColumnKeywordOpt IfExists ColumnName RestrictOrCascadeOpt | '
    ↪ PRIMARY' 'KEY' | 'PARTITION' IfExists PartitionNameList | (
    ↪ KeyOrIndex IfExists | 'CHECK' ) Identifier | 'FOREIGN' 'KEY' IfExists
    ↪  Symbol )
|   'EXCHANGE' 'PARTITION' Identifier 'WITH' 'TABLE' TableName
    ↪ WithValidationOpt
|   ( 'IMPORT' | 'DISCARD' ) ( 'PARTITION' AllOrPartitionNameList )? '
    ↪ TABLESPACE'
|   'REORGANIZE' 'PARTITION' NoWriteToBinLogAliasOpt
    ↪ ReorganizePartitionRuleOpt
|   'ORDER' 'BY' AlterOrderItem ( ',' AlterOrderItem )*
|   ( 'DISABLE' | 'ENABLE' ) 'KEYS'
|   ( 'MODIFY' ColumnKeywordOpt IfExists | 'CHANGE' ColumnKeywordOpt
    ↪ IfExists ColumnName ) ColumnDef ColumnPosition
|   'ALTER' ( ColumnKeywordOpt ColumnName ( 'SET' 'DEFAULT' ( SignedLiteral
    ↪ | '(' Expression ')' ) | 'DROP' 'DEFAULT' ) | 'CHECK' Identifier
```

```
    ↪ EnforcedOrNot | 'INDEX' Identifier IndexInvisible )
|   'RENAME' ( ( 'COLUMN' | KeyOrIndex ) Identifier 'TO' Identifier | ( 'TO'
    ↪  | '='? | 'AS' ) TableName )
|   LockClause
|   AlgorithmClause
|   'FORCE'
|   ( 'WITH' | 'WITHOUT' ) 'VALIDATION'
|   'SECONDARY_LOAD'
|   'SECONDARY_UNLOAD'
```

4.1.5.9.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 (c1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.03 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> EXPLAIN SELECT * FROM t1 WHERE c1 = 3;
+--
    ↪ ------------------+---------+------+---------------------------------------
    ↪
| id                | count   | task | operator info
    ↪                                      |
+--
    ↪ ------------------+---------+------+---------------------------------------
    ↪
| TableReader_7     | 10.00   | root | data:Selection_6
    ↪                                     |
|  ─Selection_6     | 10.00   | cop | eq(test.t1.c1, 3)
    ↪                                     |
|    ─TableScan_5   | 10000.00 | cop | table:t1, range:[-inf,+inf], keep
    ↪ order:false, stats:pseudo |
+--
    ↪ ------------------+---------+------+---------------------------------------
    ↪
3 rows in set (0.00 sec)

mysql> ALTER TABLE t1 ADD INDEX (c1);
Query OK, 0 rows affected (0.30 sec)

mysql> EXPLAIN SELECT * FROM t1 WHERE c1 = 3;
```

```
+--
    ↪ ----------------+-------+------+-----------------------------------------
    ↪
| id                | count | task | operator info
    ↪                                          |
+--
    ↪ ----------------+-------+------+-----------------------------------------
    ↪
| IndexReader_6     | 10.00 | root | index:IndexScan_5
    ↪                                         |
| └─IndexScan_5     | 10.00 | cop | table:t1, index:c1, range:[3,3], keep
    ↪ order:false, stats:pseudo |
+--
    ↪ ----------------+-------+------+-----------------------------------------
    ↪
2 rows in set (0.00 sec)
```

4.1.5.9.3 MySQL compatibility

- All of the data types except spatial types are supported.
- `FULLTEXT`, `HASH` and `SPATIAL` indexes are not supported.

4.1.5.9.4 See also

- ADD COLUMN
- DROP COLUMN
- ADD INDEX
- DROP INDEX
- RENAME INDEX
- CREATE TABLE
- DROP TABLE
- SHOW CREATE TABLE

4.1.5.10 ALTER USER

This statement changes an existing user inside the TiDB privilege system. In the MySQL privilege system, a user is the combination of a username and the host from which they are connecting from. Thus, it is possible to create a user `'newuser2'@'192.168.1.1'` who is only able to connect from the IP address `192.168.1.1`. It is also possible to have two users have the same user-portion, and different permissions as they login from different hosts.

4.1.5.10.1 Synopsis

```
AlterUserStmt ::=
    'ALTER' 'USER' IfExists (UserSpecList RequireClauseOpt ConnectionOptions
       ↪  PasswordOrLockOptions | 'USER' '(' ')' 'IDENTIFIED' 'BY'
       ↪ AuthString)

UserSpecList ::=
    UserSpec ( ',' UserSpec )*

UserSpec ::=
    Username AuthOption
```

4.1.5.10.2 Examples

```
mysql> CREATE USER 'newuser' IDENTIFIED BY 'newuserpassword';
Query OK, 1 row affected (0.01 sec)

mysql> SHOW CREATE USER 'newuser';
+--
   ↪ ------------------------------------------------------------------------
   ↪
| CREATE USER for newuser@%
   ↪
   ↪ |
+--
   ↪ ------------------------------------------------------------------------
   ↪
| CREATE USER 'newuser'@'%' IDENTIFIED WITH 'mysql_native_password' AS '
   ↪ *5806E04BBEE79E1899964C6A04D68BCA69B1A879' REQUIRE NONE PASSWORD
   ↪ EXPIRE DEFAULT ACCOUNT UNLOCK |
+--
   ↪ ------------------------------------------------------------------------
   ↪
1 row in set (0.00 sec)

mysql> ALTER USER 'newuser' IDENTIFIED BY 'newnewpassword';
Query OK, 0 rows affected (0.02 sec)

mysql> SHOW CREATE USER 'newuser';
+--
   ↪ ------------------------------------------------------------------------
   ↪
| CREATE USER for newuser@%
   ↪
```

```
    ↪ |
+--
    ↪ ----------------------------------------------------------------------------
    ↪
| CREATE USER 'newuser'@'%' IDENTIFIED WITH 'mysql_native_password' AS '*
    ↪ FB8A1EA1353E8775CA836233E367FBDFCB37BE73' REQUIRE NONE PASSWORD
    ↪ EXPIRE DEFAULT ACCOUNT UNLOCK |
+--
    ↪ ----------------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)
```

4.1.5.10.3 MySQL compatibility

- In MySQL this statement is used to change attributes such as to expire a password. This functionality is not yet supported by TiDB.

4.1.5.10.4 See also

- Security Compatibility with MySQL
- CREATE USER
- DROP USER
- SHOW CREATE USER

4.1.5.11 ANALYZE TABLE

This statement updates the statistics that TiDB builds on tables and indexes. It is recommended to run `ANALYZE TABLE` after performing a large batch update or import of records, or when you notice that query execution plans are sub-optimal.

TiDB will also automatically update its statistics over time as it discovers that they are inconsistent with its own estimates.

4.1.5.11.1 Synopsis

```
AnalyzeTableStmt ::=
    'ANALYZE' ( 'TABLE' ( TableNameList | TableName ( 'INDEX' IndexNameList
        ↪ | 'PARTITION' PartitionNameList ( 'INDEX' IndexNameList )? ) ) | '
        ↪ INCREMENTAL' 'TABLE' TableName ( 'PARTITION' PartitionNameList )?
        ↪ 'INDEX' IndexNameList ) AnalyzeOptionListOpt

TableNameList ::=
    TableName (',' TableName)*
```

```
TableName ::=
    Identifier ( '.' Identifier )?
```

4.1.5.11.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 (c1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.03 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE t1 ADD INDEX (c1);
Query OK, 0 rows affected (0.30 sec)

mysql> EXPLAIN SELECT * FROM t1 WHERE c1 = 3;
+--
    ↪ ----------------+-------+------+-----------------------------------------
    ↪
| id              | count | task | operator info
    ↪                                        |
+--
    ↪ ----------------+-------+------+-----------------------------------------
    ↪
| IndexReader_6   | 10.00 | root | index:IndexScan_5
    ↪                                       |
| └─IndexScan_5   | 10.00 | cop | table:t1, index:c1, range:[3,3], keep
    ↪ order:false, stats:pseudo |
+--
    ↪ ----------------+-------+------+-----------------------------------------
    ↪
2 rows in set (0.00 sec)

mysql> analyze table t1;
Query OK, 0 rows affected (0.13 sec)

mysql> EXPLAIN SELECT * FROM t1 WHERE c1 = 3;
+--
    ↪ ----------------+-------+------+-----------------------------------------
    ↪
| id              | count | task | operator info
    ↪                                  |
+--
```

```
    ↪ ----------------+-------+------+-----------------------------------------------
    ↪
| IndexReader_6   | 1.00 | root | index:IndexScan_5
    ↪                              |
|  ─IndexScan_5   | 1.00 | cop  | table:t1, index:c1, range:[3,3], keep
    ↪ order:false |
+--
    ↪ ----------------+-------+------+-----------------------------------------------
    ↪
2 rows in set (0.00 sec)
```

4.1.5.11.3 MySQL compatibility

This statement is syntactically similar with MySQL. However, `ANALYZE TABLE` may take significantly longer to execute on TiDB, as internally it operates in a different manner.

4.1.5.11.4 See also

- EXPLAIN
- EXPLAIN ANALYZE

4.1.5.12 BEGIN

This statement starts a new transaction inside of TiDB. It is similar to the statements `START TRANSACTION` and `SET autocommit=0`.

In the absence of a `BEGIN` statement, every statement will by default autocommit in its own transaction. This behavior ensures MySQL compatibility.

4.1.5.12.1 Synopsis

BeginTransactionStmt:

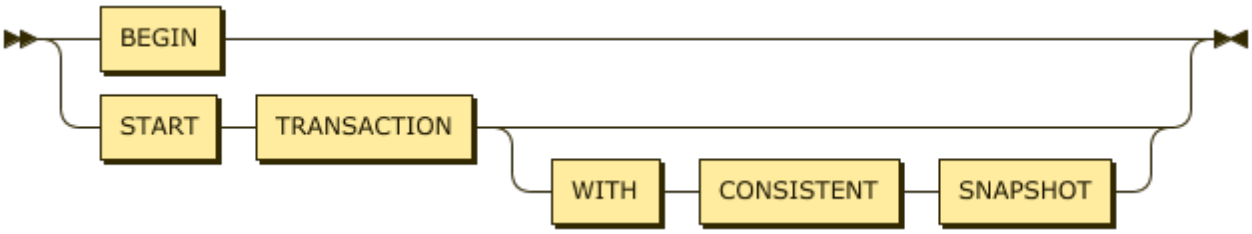

Figure 37: BeginTransactionStmt

4.1.5.12.2 Examples

```
mysql> CREATE TABLE t1 (a int NOT NULL PRIMARY KEY);
Query OK, 0 rows affected (0.12 sec)

mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t1 VALUES (1);
Query OK, 1 row affected (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)
```

4.1.5.12.3 MySQL compatibility

TiDB supports the syntax extension of `BEGIN PESSIMISTIC` or `BEGIN OPTIMISTIC`. This enables you to override the default transactional model for your transaction.

4.1.5.12.4 See also

- COMMIT
- ROLLBACK
- START TRANSACTION

4.1.5.13 CHANGE COLUMN

The `ALTER TABLE.. CHANGE COLUMN` statement changes a column on an existing table. The change can include both renaming the column, and changing the data type to a compatible type.

4.1.5.13.1 Synopsis

AlterTableStmt:

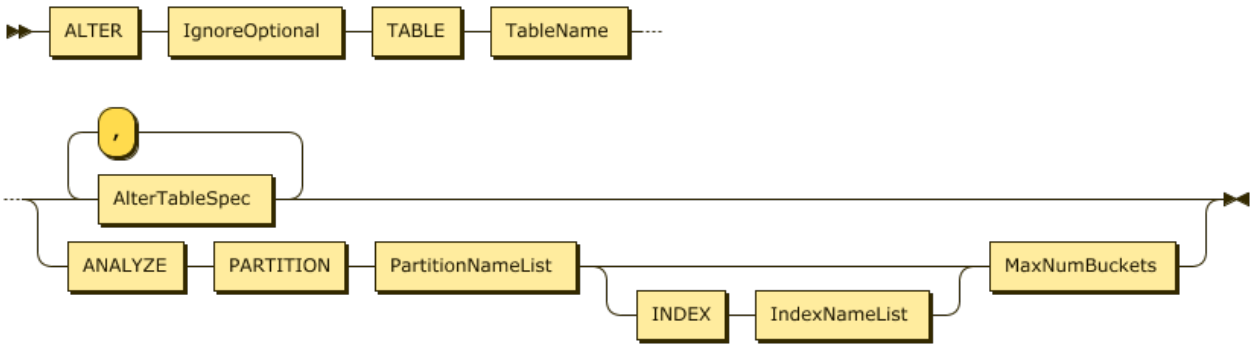

Figure 38: AlterTableStmt

AlterTableSpec:

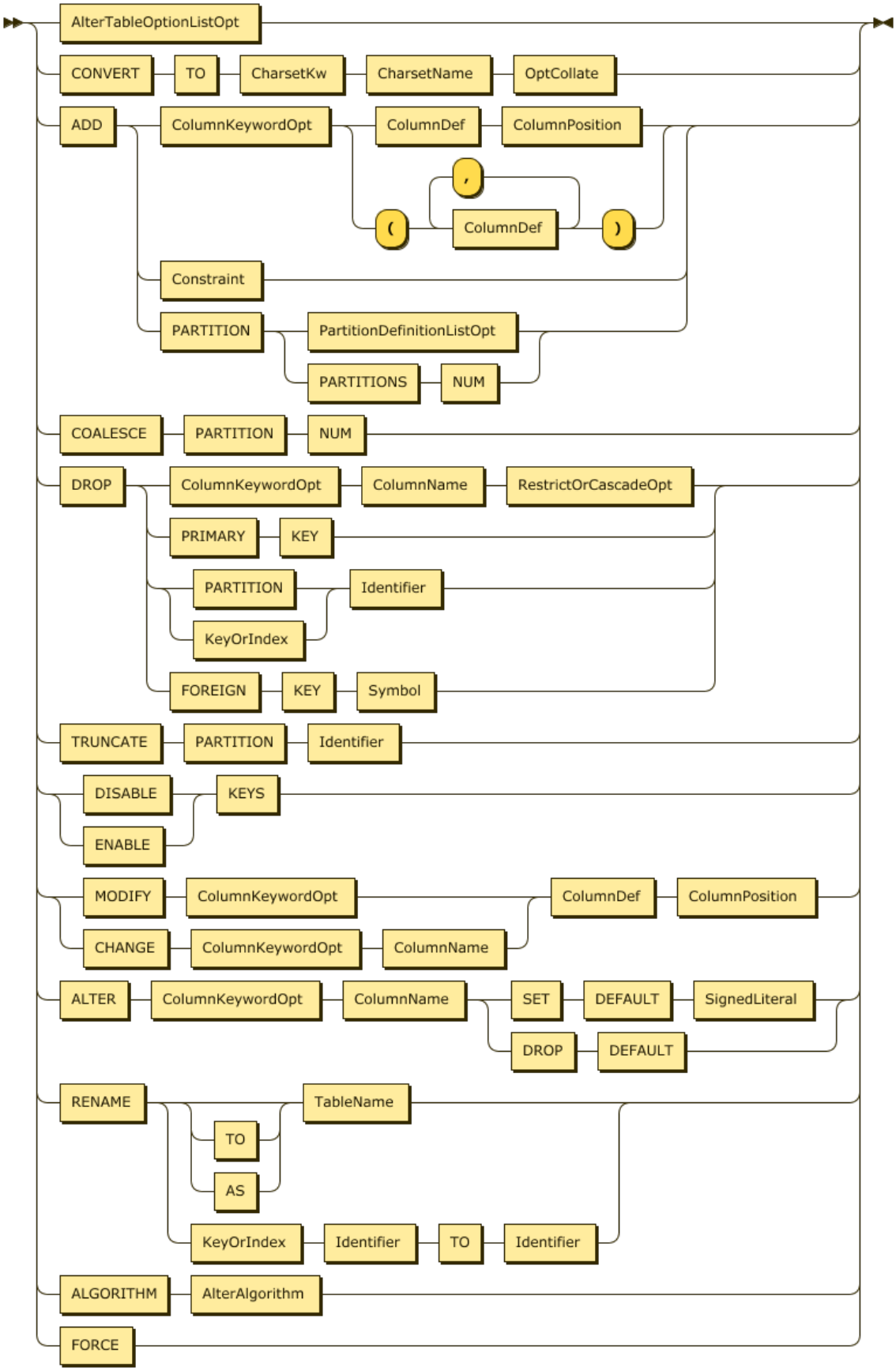

Figure 39: AlterTableSpec

ColumnKeywordOpt:

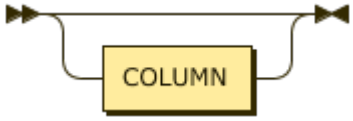

Figure 40: ColumnKeywordOpt

ColumnName:

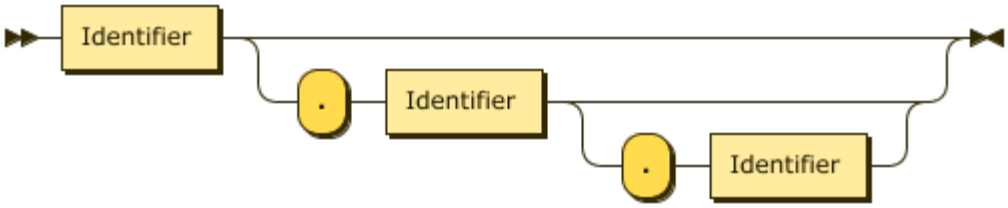

Figure 41: ColumnName

ColumnDef:

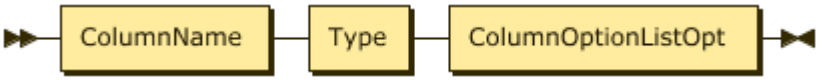

Figure 42: ColumnDef

ColumnPosition:

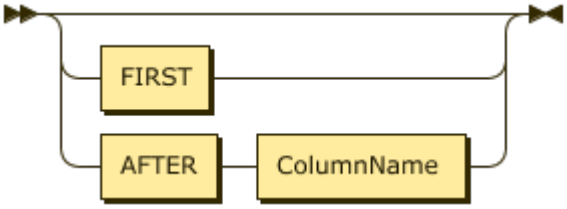

Figure 43: ColumnPosition

4.1.5.13.2 Examples

```
mysql> CREATE TABLE t1 (id int not null primary key AUTO_INCREMENT, col1
    ↪ INT);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 (col1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0
```

```
mysql>
mysql> ALTER TABLE t1 CHANGE col1 col2 INT;
Query OK, 0 rows affected (0.09 sec)

mysql> ALTER TABLE t1 CHANGE col2 col3 BIGINT, ALGORITHM=INSTANT;
Query OK, 0 rows affected (0.08 sec)

mysql>
mysql> ALTER TABLE t1 CHANGE col3 col3 INT;
ERROR 1105 (HY000): unsupported modify column length 11 is less than origin
    ↪  20
mysql> ALTER TABLE t1 CHANGE col3 col3 BLOB;
ERROR 1105 (HY000): unsupported modify column type 252 not match origin 8
mysql> ALTER TABLE t1 CHANGE col3 col4 BIGINT, CHANGE id id2 INT NOT NULL;
ERROR 1105 (HY000): can't run multi schema change
```

4.1.5.13.3 MySQL compatibility

- Making multiple changes in a single `ALTER TABLE` statement is not currently supported.
- Only certain types of data type changes are supported. For example, an `INTEGER` to `BIGINT` is supported, but the reverse is not possible. Changing from an integer to a string format or blob is not supported.

4.1.5.13.4 See also

- CREATE TABLE
- SHOW CREATE TABLE
- ADD COLUMN
- DROP COLUMN
- MODIFY COLUMN

4.1.5.14 COMMIT

This statement commits a transaction inside of the TIDB server.

In the absence of a `BEGIN` or `START TRANSACTION` statement, the default behavior of TiDB is that every statement will be its own transaction and autocommit. This behavior ensures MySQL compatibility.

4.1.5.14.1 Synopsis

CommitStmt:

Figure 44: CommitStmt

4.1.5.14.2 Examples

```
mysql> CREATE TABLE t1 (a int NOT NULL PRIMARY KEY);
Query OK, 0 rows affected (0.12 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t1 VALUES (1);
Query OK, 1 row affected (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)
```

4.1.5.14.3 MySQL compatibility

- By default, TiDB 3.0.8 and later versions use Pessimistic Locking. When using Optimistic Locking, it is important to consider that a `COMMIT` statement might fail because rows have been modified by another transaction.
- When Optimistic Locking is enabled, `UNIQUE` and `PRIMARY KEY` constraint checks are deferred until statement commit. This results in additional situations where a `COMMIT` statement might fail. This behavior can be changed by setting `tidb_constraint_check_in_place=TRUE`.

4.1.5.14.4 See also

- START TRANSACTION
- ROLLBACK
- BEGIN
- Lazy checking of constraints

4.1.5.15 CREATE DATABASE

This statement creates a new database in TiDB. The MySQL terminology for 'database' most closely maps to a schema in the SQL standard.

4.1.5.15.1 Synopsis

CreateDatabaseStmt:

Figure 45: CreateDatabaseStmt

DatabaseSym:

Figure 46: DatabaseSym

IfNotExists:

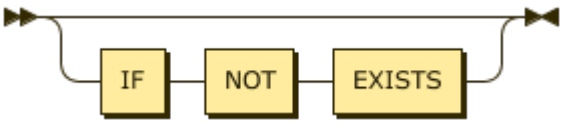

Figure 47: IfNotExists

DBName:

Figure 48: DBName

DatabaseOptionListOpt:

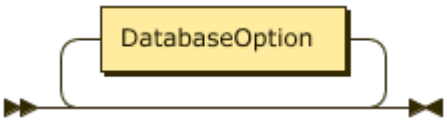

Figure 49: DatabaseOptionListOpt

4.1.5.15.2 Syntax

The `CREATE DATABASE` statement is used to create a database, and to specify the default properties of the database, such as the default character set and collation. `CREATE SCHEMA` is a synonym for `CREATE DATABASE`.

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_specification] ...

create_specification:
    [DEFAULT] CHARACTER SET [=] charset_name
  | [DEFAULT] COLLATE [=] collation_name
```

If you create an existing database and does not specify `IF NOT EXISTS`, an error is displayed.

The `create_specification` option is used to specify the specific `CHARACTER SET` and `COLLATE` in the database. Currently, TiDB only supports some of the character sets and collations. For details, see Character Set Support.

4.1.5.15.3 Examples

```
mysql> CREATE DATABASE mynewdatabase;
Query OK, 0 rows affected (0.09 sec)

mysql> USE mynewdatabase;
Database changed
mysql> CREATE TABLE t1 (a int);
Query OK, 0 rows affected (0.11 sec)

mysql> SHOW TABLES;
+------------------------+
| Tables_in_mynewdatabase |
+------------------------+
| t1                     |
+------------------------+
1 row in set (0.00 sec)
```

4.1.5.15.4 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.15.5 See also

- USE
- ALTER DATABASE
- DROP DATABASE
- SHOW DATABASES

4.1.5.16 CREATE INDEX

This statement adds a new index to an existing table. It is an alternative syntax to `ALTER TABLE .. ADD INDEX`, and included for MySQL compatibility.

4.1.5.16.1 Synopsis

CreateIndexStmt:

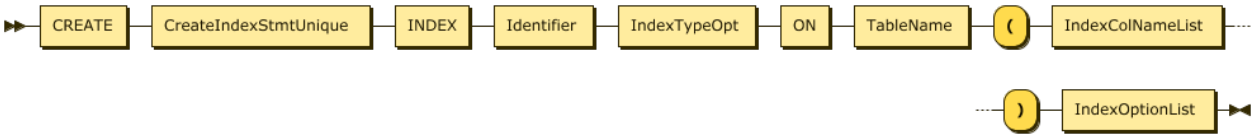

Figure 50: CreateIndexStmt

CreateIndexStmtUnique:

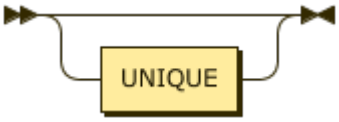

Figure 51: CreateIndexStmtUnique

Identifier:

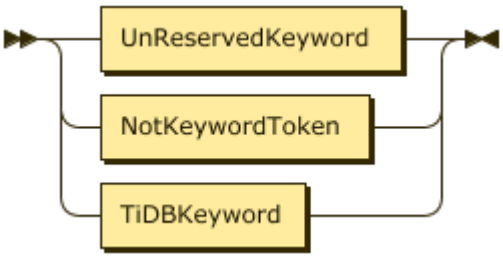

Figure 52: Identifier

IndexTypeOpt:

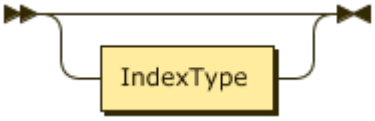

Figure 53: IndexTypeOpt

TableName:

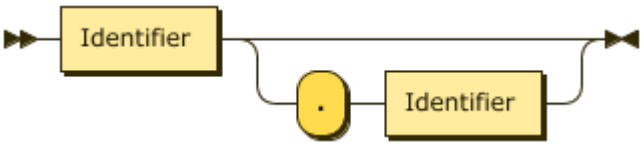

Figure 54: TableName

IndexColNameList:

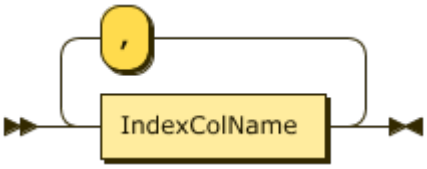

Figure 55: IndexColNameList

IndexOptionList:

Figure 56: IndexOptionList

IndexOption:

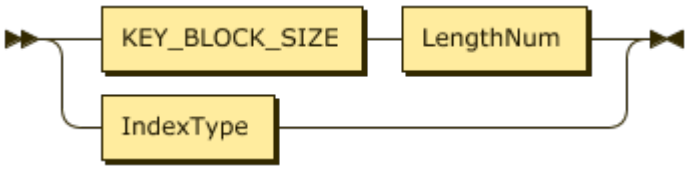

Figure 57: IndexOption

4.1.5.16.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.10 sec)

mysql> INSERT INTO t1 (c1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> EXPLAIN SELECT * FROM t1 WHERE c1 = 3;
```

```
+--
    ↪ -------------------+-----------+-------+----------------------------------------------
    ↪
| id                  | count    | task | operator info
    ↪                                          |
+--
    ↪ -------------------+-----------+-------+----------------------------------------------
    ↪
| TableReader_7      | 10.00    | root | data:Selection_6
    ↪                                       |
|  ─Selection_6      | 10.00    | cop  | eq(test.t1.c1, 3)
    ↪                                       |
|    ─TableScan_5    | 10000.00 | cop  | table:t1, range:[-inf,+inf], keep
    ↪ order:false, stats:pseudo |
+--
    ↪ -------------------+-----------+-------+----------------------------------------------
    ↪
3 rows in set (0.00 sec)

mysql> CREATE INDEX c1 ON t1 (c1);
Query OK, 0 rows affected (0.30 sec)

mysql> EXPLAIN SELECT * FROM t1 WHERE c1 = 3;
+--
    ↪ -----------------+--------+-------+-----------------------------------------------------
    ↪
| id                | count | task | operator info
    ↪                                             |
+--
    ↪ -----------------+--------+-------+-----------------------------------------------------
    ↪
| IndexReader_6    | 10.00 | root | index:IndexScan_5
    ↪                                          |
|  ─IndexScan_5    | 10.00 | cop  | table:t1, index:c1, range:[3,3], keep
    ↪ order:false, stats:pseudo |
+--
    ↪ -----------------+--------+-------+-----------------------------------------------------
    ↪
2 rows in set (0.00 sec)

mysql> ALTER TABLE t1 DROP INDEX c1;
Query OK, 0 rows affected (0.30 sec)

mysql> CREATE UNIQUE INDEX c1 ON t1 (c1);
Query OK, 0 rows affected (0.31 sec)
```

4.1.5.16.3 Associated session variables

The global variables associated with the `CREATE INDEX` statement are `tidb_ddl_reorg_worker_cnt` , `tidb_ddl_reorg_batch_size` and `tidb_ddl_reorg_priority`. Refer to TiDB-specific system variables for details.

4.1.5.16.4 MySQL compatibility

- `FULLTEXT`, `HASH` and `SPATIAL` indexes are not supported.
- Descending indexes are not supported (similar to MySQL 5.7).
- Adding the primary key constraint to a table is not supported by default. You can enable the feature by setting the `alter-primary-key` configuration item to `true`. For details, see alter-primary-key.

4.1.5.16.5 See also

- ADD INDEX
- DROP INDEX
- RENAME INDEX
- ADD COLUMN
- CREATE TABLE
- EXPLAIN

4.1.5.17 CREATE ROLE

This statement creates a new role, which can be assigned to users as part of role-based access control.

4.1.5.17.1 Synopsis

CreateRoleStmt:

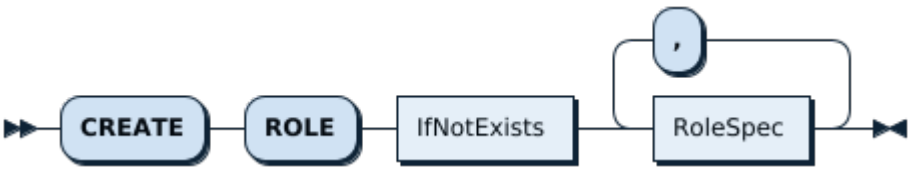

Figure 58: CreateRoleStmt

IfNotExists:

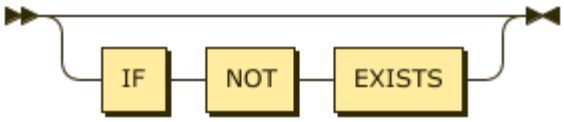

Figure 59: IfNotExists

RoleSpec:

Figure 60: RoleSpec

4.1.5.17.2 Examples

Create a new role for the analytics team, and a new user called `jennifer`:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 37
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> CREATE ROLE analyticsteam;
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT SELECT ON test.* TO analyticsteam;
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE USER jennifer;
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT analyticsteam TO jennifer;
Query OK, 0 rows affected (0.01 sec)
```

Note that by default `jennifer` needs to `SET ROLE analyticsteam` in order to be able to use the privileges associated with the role:

```
$ mysql -ujennifer
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SHOW GRANTS;
+----------------------------------------------+
| Grants for User                              |
+----------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'  |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+----------------------------------------------+
2 rows in set (0.00 sec)

mysql> SHOW TABLES in test;
ERROR 1044 (42000): Access denied for user 'jennifer'@'%' to database 'test'
mysql> SET ROLE analyticsteam;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS;
+----------------------------------------------+
| Grants for User                              |
+----------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'  |
| GRANT Select ON test.* TO 'jennifer'@'%' |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+----------------------------------------------+
3 rows in set (0.00 sec)

mysql> SHOW TABLES IN test;
+----------------+
```

```
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)
```

The statement `SET DEFAULT ROLE` can be used to associate a role to `jennifer` so that she will not have to execute the statement `SET ROLE` in order to assume the privileges associated with the role:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 34
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SET DEFAULT ROLE analyticsteam TO jennifer;
Query OK, 0 rows affected (0.02 sec)
```

```
$ mysql -ujennifer
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 35
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.
```

```
mysql> SHOW GRANTS;
+-------------------------------------------+
| Grants for User                           |
+-------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'  |
| GRANT Select ON test.* TO 'jennifer'@'%' |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+-------------------------------------------+
3 rows in set (0.00 sec)

mysql> SHOW TABLES IN test;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)
```

4.1.5.17.3 MySQL compatibility

This statement is understood to be fully compatible with roles, which are a feature of MySQL 8.0. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.17.4 See also

- DROP ROLE
- GRANT <role>
- REVOKE <role>
- SET ROLE
- SET DEFAULT ROLE
- Role-Based Access Control

4.1.5.18 CREATE TABLE LIKE

This statement copies the definition of an existing table, without copying any data.

4.1.5.18.1 Synopsis

CreateTableStmt:

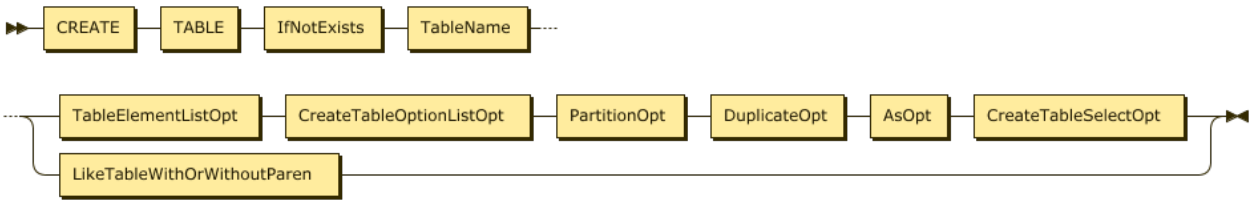

Figure 61: CreateTableStmt

LikeTableWithOrWithoutParen:

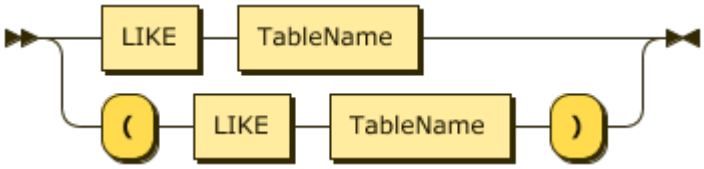

Figure 62: LikeTableWithOrWithoutParen

TableName:

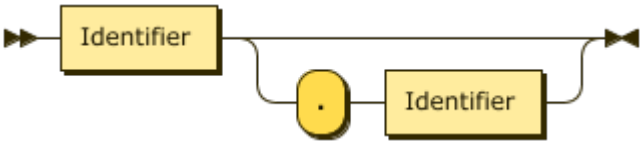

Figure 63: TableName

4.1.5.18.2 Examples

```
mysql> CREATE TABLE t1 (a INT NOT NULL);
Query OK, 0 rows affected (0.13 sec)

mysql> INSERT INTO t1 VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM t1;
+---+
| a |
+---+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
```

```
+---+
5 rows in set (0.00 sec)

mysql> CREATE TABLE t2 LIKE t1;
Query OK, 0 rows affected (0.10 sec)

mysql> SELECT * FROM t2;
Empty set (0.00 sec)
```

4.1.5.18.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.18.4 See also

- CREATE TABLE
- SHOW CREATE TABLE

4.1.5.19 CREATE TABLE

This statement creates a new table in the currently selected database. It behaves similarly to the `CREATE TABLE` statement in MySQL.

4.1.5.19.1 Synopsis

CreateTableStmt:

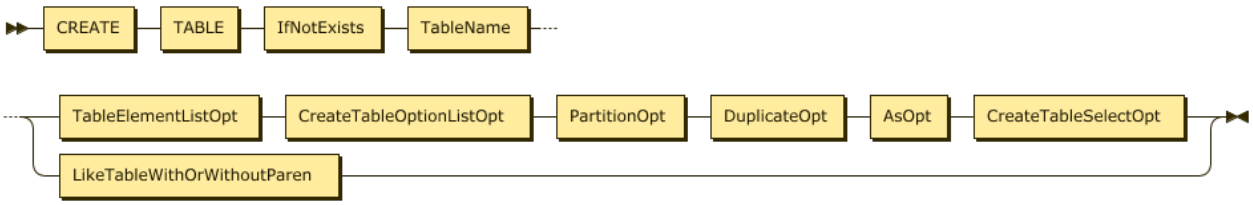

Figure 64: CreateTableStmt

IfNotExists:

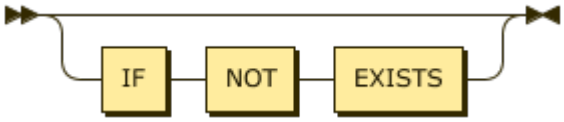

Figure 65: IfNotExists

TableName:

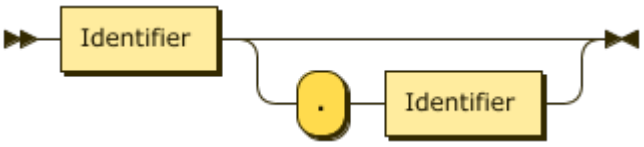

Figure 66: TableName

TableElementListOpt:

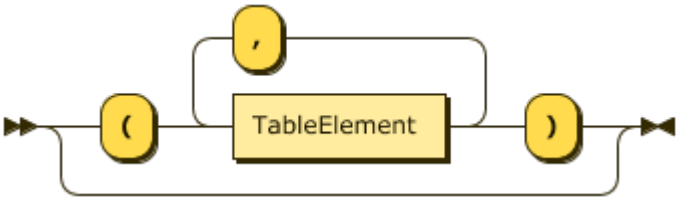

Figure 67: TableElementListOpt

TableElement:

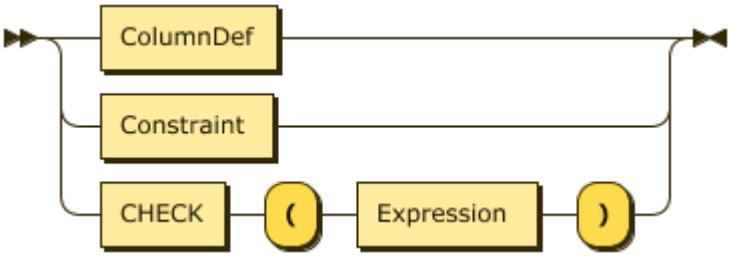

Figure 68: TableElement

PartitionOpt:

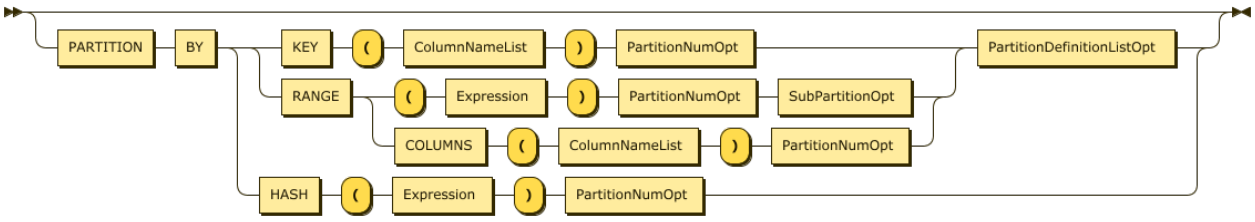

Figure 69: PartitionOpt

ColumnDef:

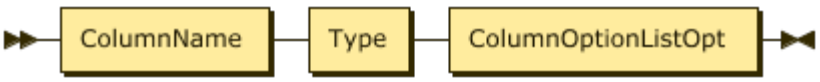

Figure 70: ColumnDef

ColumnName:

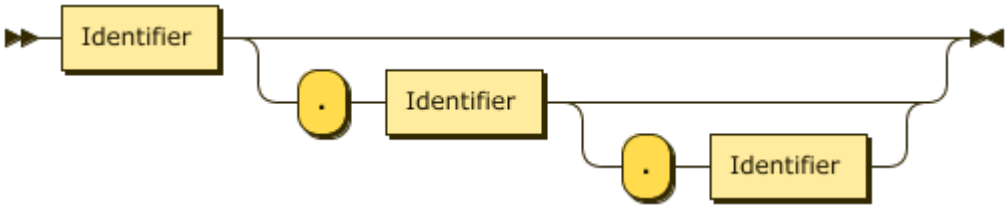

Figure 71: ColumnName

Type:

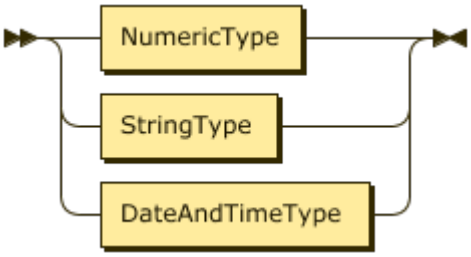

Figure 72: Type

ColumnOptionListOpt:

Figure 73: ColumnOptionListOpt

TableOptionListOpt:

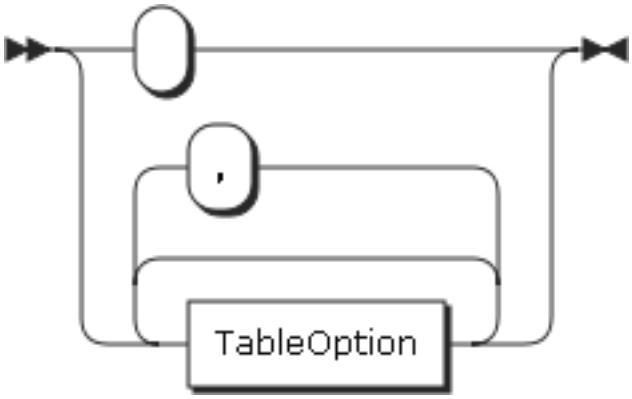

Figure 74: TableOptionListOpt

The following *table_options* are supported. Other options such as `AVG_ROW_LENGTH` ↪ , `CHECKSUM`, `COMPRESSION`, `CONNECTION`, `DELAY_KEY_WRITE`, `ENGINE`, `KEY_BLOCK_SIZE`, `MAX_ROWS`, `MIN_ROWS`, `ROW_FORMAT` and `STATS_PERSISTENT` are parsed but ignored.

Options	Description	Example
`AUTO_INCREMENT` ↪	The initial value of the increment field	`AUTO_INCREMENT` ↪ = 5
`SHARD_ROW_ID_BITS` ↪	Defines the number of bits for the implicit `_tidb_rowid` ↪ shards	`SHARD_ROW_ID_BITS` ↪ = 4
`PRE_SPLIT_REGIONS` ↪	Pre-split 2^(↪ `PRE_SPLIT_REGIONS` ↪) Regions when creating a table	`PRE_SPLIT_REGIONS` ↪ = 4
`CHARACTER` ↪ `SET`	To specify the character set for the table	`CHARACTER` ↪ ↪ `SET` = 'utf8mb4'

Options	Description	Example
COMMENT	The comment information	COMMENT ↪ = 'comment info'

Note:

The `split-table` configuration option is enabled by default. When it is enabled, a separate Region is created for each newly created table. For details, see TiDB configuration file.

4.1.5.19.2 Examples

Creating a simple table and inserting one row:

```
CREATE TABLE t1 (a int);
DESC t1;
SHOW CREATE TABLE t1\G
INSERT INTO t1 (a) VALUES (1);
SELECT * FROM t1;
```

```
mysql> drop table if exists t1;
Query OK, 0 rows affected (0.23 sec)

mysql> CREATE TABLE t1 (a int);
Query OK, 0 rows affected (0.09 sec)

mysql> DESC t1;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| a     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
1 row in set (0.00 sec)

mysql> SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `a` int(11) DEFAULT NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
1 row in set (0.00 sec)

mysql> INSERT INTO t1 (a) VALUES (1);
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM t1;
+------+
| a    |
+------+
|    1 |
+------+
1 row in set (0.00 sec)
```

Dropping a table if it exists, and conditionally creating a table if it does not exist:

```
DROP TABLE IF EXISTS t1;
CREATE TABLE IF NOT EXISTS t1 (
 id BIGINT NOT NULL PRIMARY KEY auto_increment,
 b VARCHAR(200) NOT NULL
);
DESC t1;
```

```
mysql> DROP TABLE IF EXISTS t1;
Query OK, 0 rows affected (0.22 sec)

mysql> CREATE TABLE IF NOT EXISTS t1 (
    -> id BIGINT NOT NULL PRIMARY KEY auto_increment,
    -> b VARCHAR(200) NOT NULL
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> DESC t1;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id    | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| b     | varchar(200) | NO |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
2 rows in set (0.00 sec)
```

4.1.5.19.3 MySQL compatibility

- TiDB does not support the syntax `CREATE TEMPORARY TABLE`.

- All of the data types except spatial types are supported.
- `FULLTEXT`, `HASH` and `SPATIAL` indexes are not supported.
- For compatibility, the `index_col_name` attribute supports the length option with a maximum length limit of 3072 bytes by default. The length limit can be changed through the `max-index-length` configuration option. For details, see TiDB configuration file.
- The `[ASC | DESC]` in `index_col_name` is currently parsed but ignored (MySQL 5.7 compatible behavior).
- The `COMMENT` attribute supports a maximum of 1024 characters and does not support the `WITH PARSER` option.
- TiDB supports at most 512 columns in a single table. The corresponding number limit in InnoDB is 1017, and the hard limit in MySQL is 4096.
- For partitioned tables, only Range, Hash and Range Columns (single column) are supported. For details, see partitioned table.
- `CHECK` constraints are parsed but ignored (MySQL 5.7 compatible behavior). For details, see Constraints.
- `FOREIGN KEY` constraints are parsed and stored, but not enforced by DML statements. For details, see Constraints.

4.1.5.19.4 See also

- Data Types
- DROP TABLE
- CREATE TABLE LIKE
- SHOW CREATE TABLE

4.1.5.20 CREATE USER

This statement creates a new user, specified with a password. In the MySQL privilege system, a user is the combination of a username and the host from which they are connecting from. Thus, it is possible to create a user `'newuser2'@'192.168.1.1'` who is only able to connect from the IP address `192.168.1.1`. It is also possible to have two users have the same user-portion, and different permissions as they login from different hosts.

4.1.5.20.1 Synopsis

CreateUserStmt:

Figure 75: CreateUserStmt

IfNotExists:

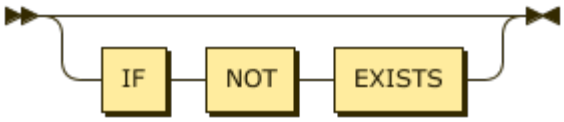

Figure 76: IfNotExists

UserSpecList:

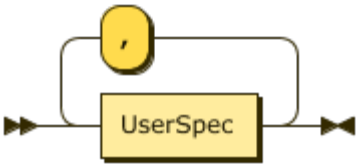

Figure 77: UserSpecList

UserSpec:

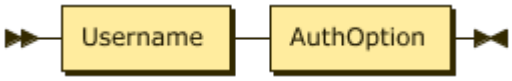

Figure 78: UserSpec

AuthOption:

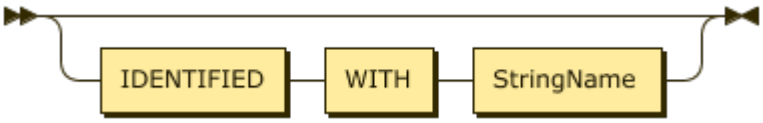

Figure 79: AuthOption

StringName:

Figure 80: StringName

4.1.5.20.2 Examples

```
mysql> CREATE USER 'newuser' IDENTIFIED BY 'newuserpassword';
Query OK, 1 row affected (0.04 sec)

mysql> CREATE USER 'newuser2'@'192.168.1.1' IDENTIFIED BY 'newuserpassword'
    ↪ ;
Query OK, 1 row affected (0.02 sec)
```

4.1.5.20.3 MySQL compatibility

- Several of the `CREATE` options are not yet supported by TiDB, and will be parsed but ignored.

4.1.5.20.4 See also

- Security Compatibility with MySQL
- DROP USER
- SHOW CREATE USER
- ALTER USER
- Privilege Management

4.1.5.21 CREATE VIEW

The `CREATE VIEW` statement saves a `SELECT` statement as a queryable object, similar to a table. Views in TiDB are non-materialized. This means that as a view is queried, TiDB will internally rewrite the query to combine the view definition with the SQL query.

4.1.5.21.1 Synopsis

CreateViewStmt:

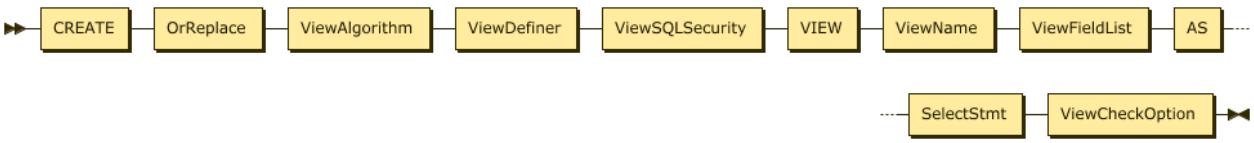

Figure 81: CreateViewStmt

OrReplace:

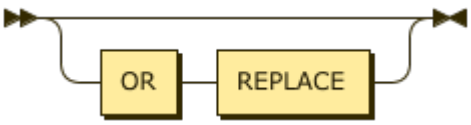

Figure 82: OrReplace

ViewAlgorithm:

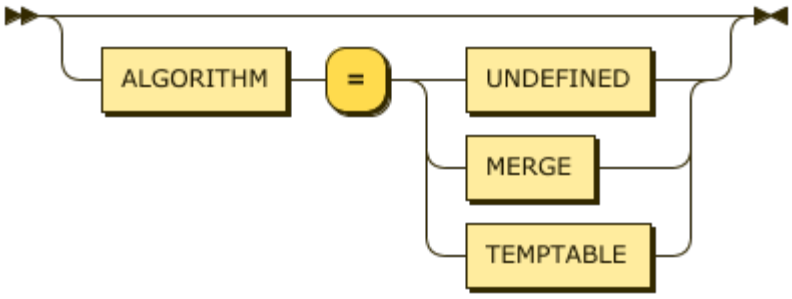

Figure 83: ViewAlgorithm

ViewDefiner:

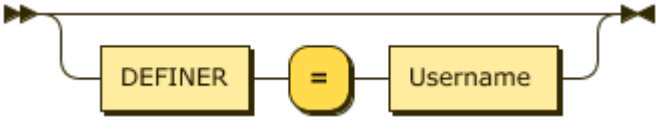

Figure 84: ViewDefiner

ViewSQLSecurity:

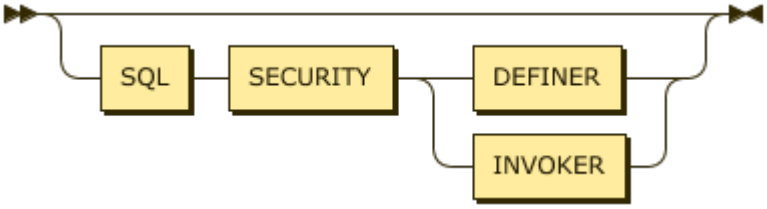

Figure 85: ViewSQLSecurity

ViewName:

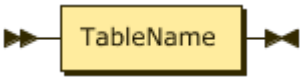

Figure 86: ViewName

ViewFieldList:

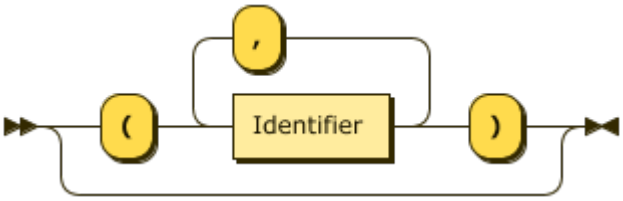

Figure 87: ViewFieldList

ViewCheckOption:

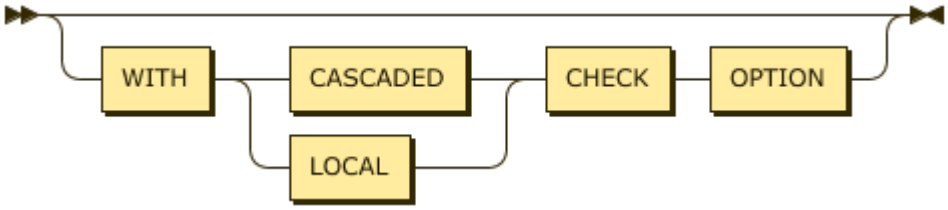

Figure 88: ViewCheckOption

4.1.5.21.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 (c1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.03 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> CREATE VIEW v1 AS SELECT * FROM t1 WHERE c1 > 2;
Query OK, 0 rows affected (0.11 sec)

mysql> SELECT * FROM t1;
+----+----+
| id | c1 |
+----+----+
|  1 |  1 |
|  2 |  2 |
|  3 |  3 |
|  4 |  4 |
|  5 |  5 |
+----+----+
5 rows in set (0.00 sec)

mysql> SELECT * FROM v1;
+----+----+
| id | c1 |
+----+----+
|  3 |  3 |
|  4 |  4 |
|  5 |  5 |
+----+----+
3 rows in set (0.00 sec)
```

```
mysql> INSERT INTO t1 (c1) VALUES (6);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM v1;
+----+----+
| id | c1 |
+----+----+
|  3 |  3 |
|  4 |  4 |
|  5 |  5 |
|  6 |  6 |
+----+----+
4 rows in set (0.00 sec)

mysql> INSERT INTO v1 (c1) VALUES (7);
ERROR 1105 (HY000): insert into view v1 is not supported now.
```

4.1.5.21.3 MySQL compatibility

- Views in TiDB are not currently insertable or updatable.

4.1.5.21.4 See also

- CREATE TABLE
- SHOW CREATE TABLE
- DROP TABLE

4.1.5.22 DEALLOCATE

The `DEALLOCATE` statement provides an SQL interface to server-side prepared statements.

4.1.5.22.1 Synopsis

DeallocateStmt:

Figure 89: DeallocateStmt

DeallocateSym:

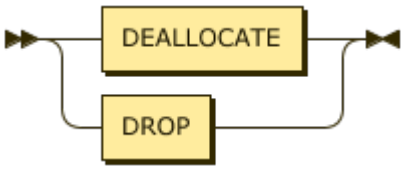

Figure 90: DeallocateSym

Identifier:

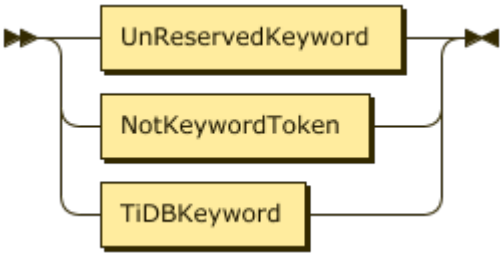

Figure 91: Identifier

4.1.5.22.2 Examples

```
mysql> PREPARE mystmt FROM 'SELECT ? as num FROM DUAL';
Query OK, 0 rows affected (0.00 sec)

mysql> SET @number = 5;
Query OK, 0 rows affected (0.00 sec)

mysql> EXECUTE mystmt USING @number;
+------+
| num |
+------+
| 5   |
+------+
1 row in set (0.00 sec)

mysql> DEALLOCATE PREPARE mystmt;
Query OK, 0 rows affected (0.00 sec)
```

4.1.5.22.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.22.4 See also

- PREPARE
- EXECUTE

4.1.5.23 DELETE

The `DELETE` statement removes rows from a specified table.

4.1.5.23.1 Synopsis

DeleteFromStmt:

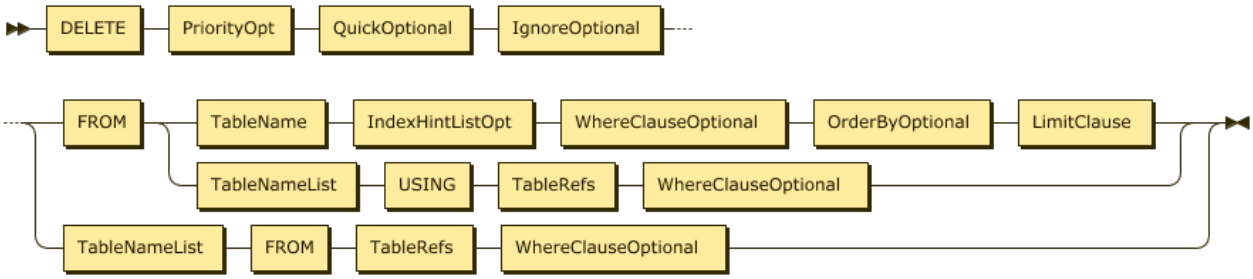

Figure 92: DeleteFromStmt

4.1.5.23.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 (c1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.03 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM t1;
+----+----+
| id | c1 |
+----+----+
|  1 |  1 |
|  2 |  2 |
|  3 |  3 |
|  4 |  4 |
|  5 |  5 |
+----+----+
5 rows in set (0.00 sec)

mysql> DELETE FROM t1 WHERE id = 4;
Query OK, 1 row affected (0.02 sec)
```

```
mysql> SELECT * FROM t1;
+----+----+
| id | c1 |
+----+----+
|  1 |  1 |
|  2 |  2 |
|  3 |  3 |
|  5 |  5 |
+----+----+
4 rows in set (0.00 sec)
```

4.1.5.23.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.23.4 See also

- INSERT
- SELECT
- UPDATE
- REPLACE

4.1.5.24 DESC

This statement is an alias to `EXPLAIN`. It is included for compatibility with MySQL.

4.1.5.25 DESCRIBE

This statement is an alias to `EXPLAIN`. It is included for compatibility with MySQL.

4.1.5.26 DO

This statement executes an expression, without returning a result. In MySQL, a common use-case is to execute stored programs without needing to handle the result. Since TiDB does not provide stored routines, this function has a more limited use.

4.1.5.26.1 Synopsis

DoStmt:

Figure 93: DoStmt

ExpressionList:

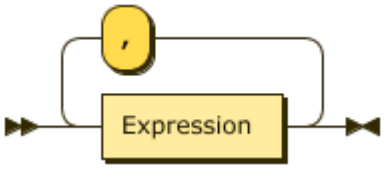

Figure 94: ExpressionList

Expression:

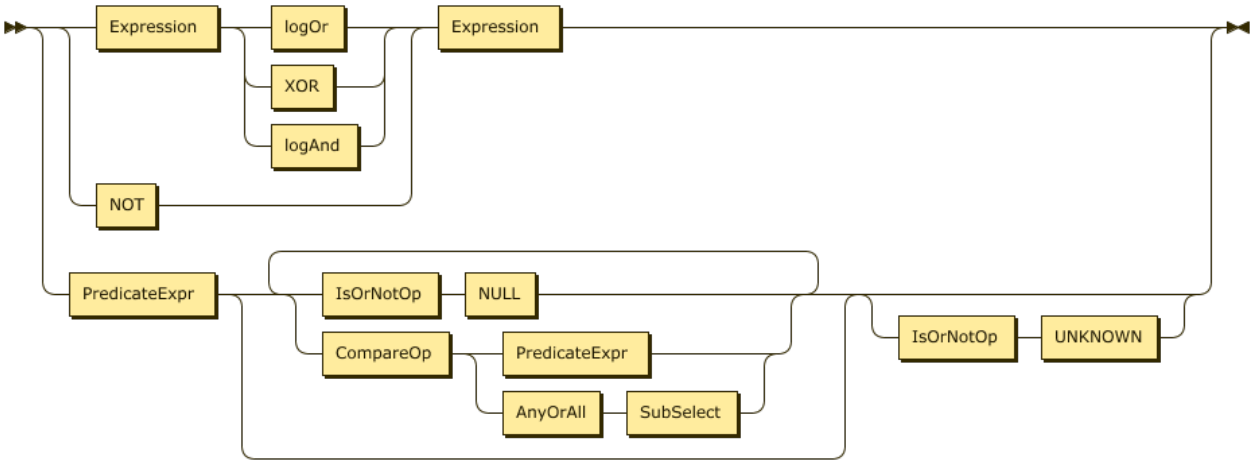

Figure 95: Expression

4.1.5.26.2 Examples

```
mysql> SELECT SLEEP(5);
+----------+
| SLEEP(5) |
+----------+
|        0 |
+----------+
1 row in set (5.00 sec)

mysql> DO SLEEP(5);
Query OK, 0 rows affected (5.00 sec)

mysql> DO SLEEP(1), SLEEP(1.5);
Query OK, 0 rows affected (2.50 sec)
```

4.1.5.26.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.26.4 See also

- SELECT

4.1.5.27 DROP COLUMN

This statement drops a column from a specified table. `DROP COLUMN` is online in TiDB, which means that it does not block read or write operations.

4.1.5.27.1 Synopsis

AlterTableStmt:

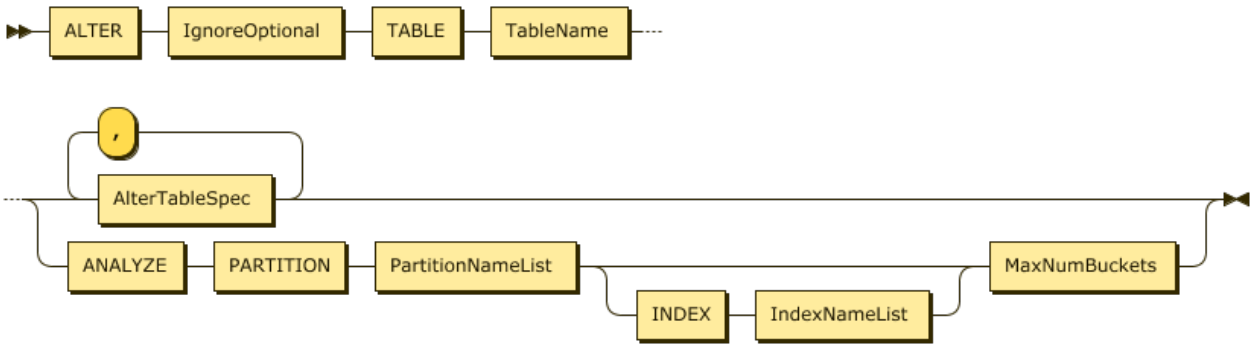

Figure 96: AlterTableStmt

AlterTableSpec:

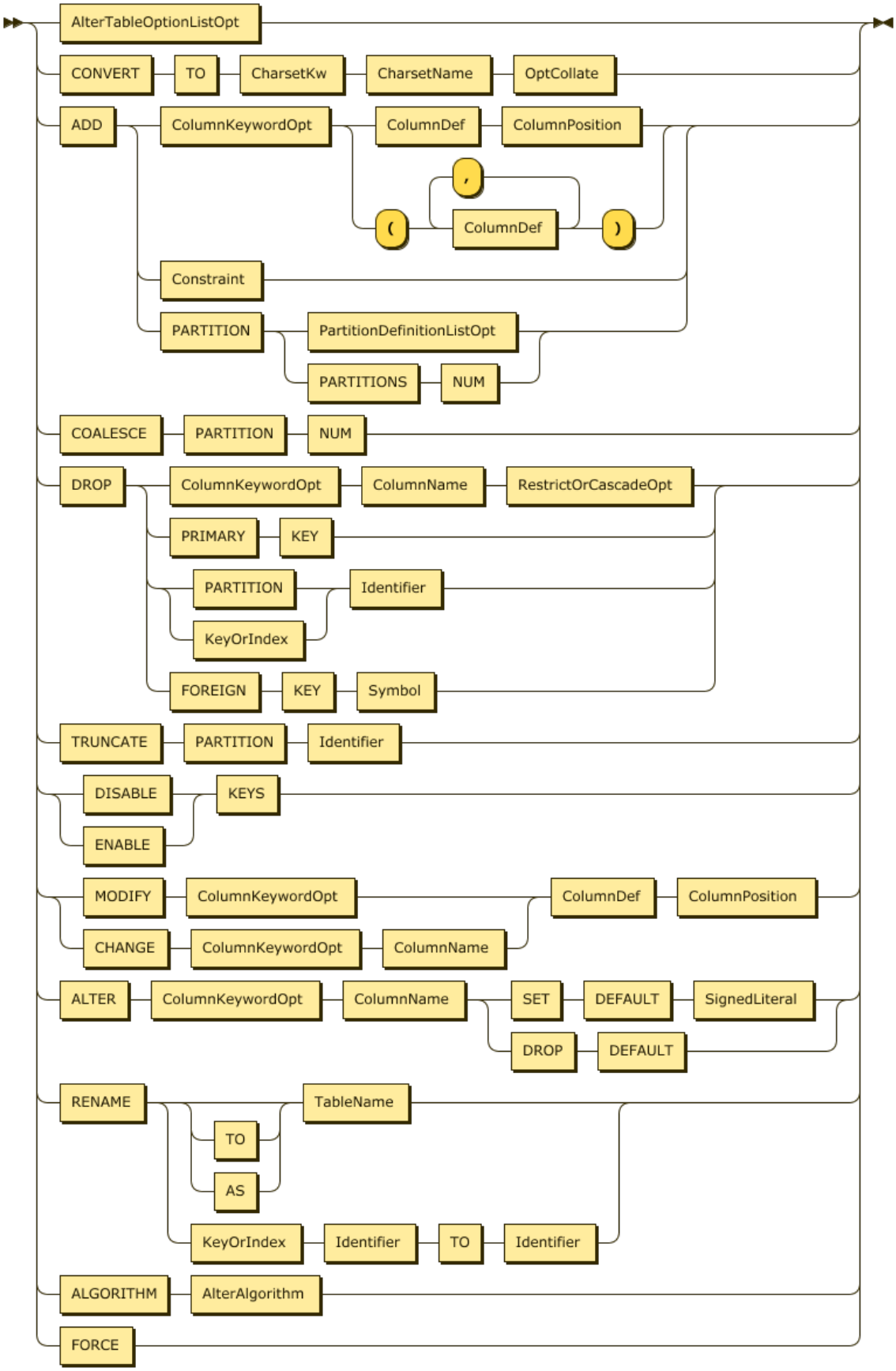

Figure 97: AlterTableSpec

ColumnKeywordOpt:

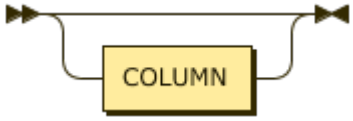

Figure 98: ColumnKeywordOpt

ColumnName:

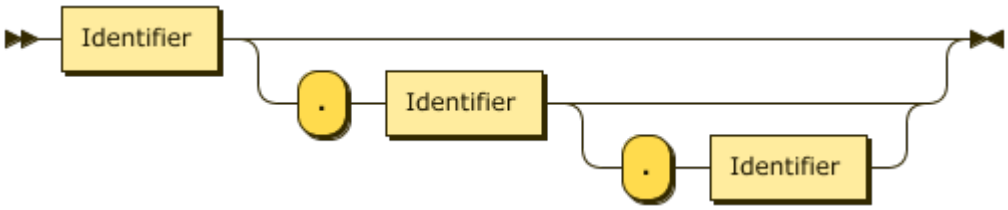

Figure 99: ColumnName

4.1.5.27.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, col1
    ↪ INT NOT NULL, col2 INT NOT NULL);
Query OK, 0 rows affected (0.12 sec)

mysql> INSERT INTO t1 (col1,col2) VALUES (1,1),(2,2),(3,3),(4,4),(5,5);
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM t1;
+----+------+------+
| id | col1 | col2 |
+----+------+------+
|  1 |    1 |    1 |
|  2 |    2 |    2 |
|  3 |    3 |    3 |
|  4 |    4 |    4 |
|  5 |    5 |    5 |
+----+------+------+
5 rows in set (0.01 sec)

mysql> ALTER TABLE t1 DROP COLUMN col1, DROP COLUMN col2;
ERROR 1105 (HY000): can't run multi schema change
mysql> SELECT * FROM t1;
+----+------+------+
| id | col1 | col2 |
```

```
+----+------+------+
|  1 |    1 |    1 |
|  2 |    2 |    2 |
|  3 |    3 |    3 |
|  4 |    4 |    4 |
|  5 |    5 |    5 |
+----+------+------+
5 rows in set (0.00 sec)

mysql> ALTER TABLE t1 DROP COLUMN col1;
Query OK, 0 rows affected (0.27 sec)

mysql> SELECT * FROM t1;
+----+------+
| id | col2 |
+----+------+
|  1 |    1 |
|  2 |    2 |
|  3 |    3 |
|  4 |    4 |
|  5 |    5 |
+----+------+
5 rows in set (0.00 sec)
```

4.1.5.27.3 MySQL compatibility

- Dropping multiple columns in the same statement is not supported.

4.1.5.27.4 See also

- ADD COLUMN
- SHOW CREATE TABLE
- CREATE TABLE

4.1.5.28 DROP DATABASE

The `DROP DATABASE` statement permanently removes a specified database schema, and all of the tables and views that were created inside. User privileges that are associated with the dropped database remain unaffected.

4.1.5.28.1 Synopsis

DropDatabaseStmt:

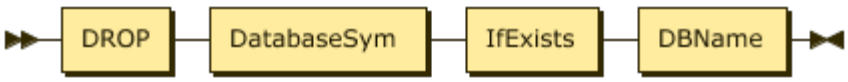

Figure 100: DropDatabaseStmt

DatabaseSym:

Figure 101: DatabaseSym

IfExists:

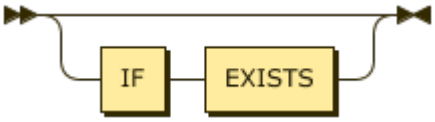

Figure 102: IfExists

DBName:

Figure 103: DBName

4.1.5.28.2 Examples

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| INFORMATION_SCHEMA |
| PERFORMANCE_SCHEMA |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql> DROP DATABASE test;
Query OK, 0 rows affected (0.25 sec)

mysql> SHOW DATABASES;
+--------------------+
```

```
| Database           |
+--------------------+
| INFORMATION_SCHEMA |
| PERFORMANCE_SCHEMA |
| mysql              |
+--------------------+
3 rows in set (0.00 sec)
```

4.1.5.28.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.28.4 See also

- CREATE DATABASE
- ALTER DATABASE

4.1.5.29 DROP INDEX

This statement removes an index from a specified table, marking space as free in TiKV.

4.1.5.29.1 Synopsis

AlterTableStmt:

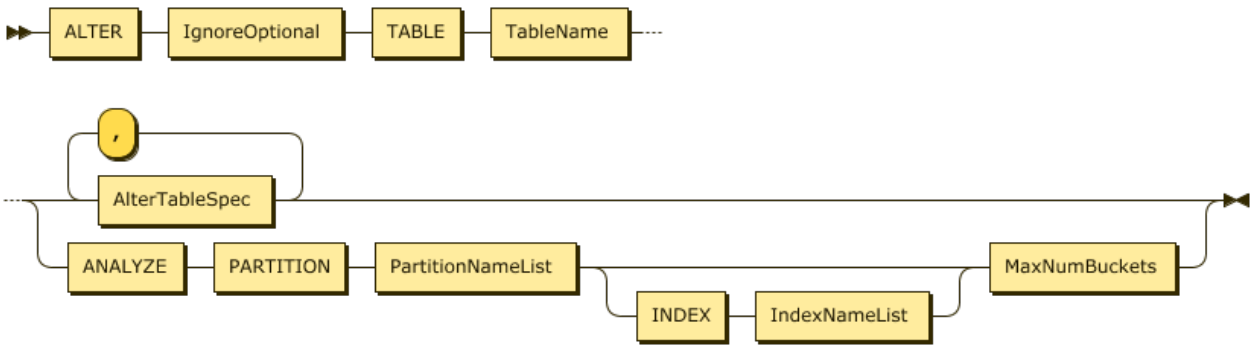

Figure 104: AlterTableStmt

AlterTableSpec:

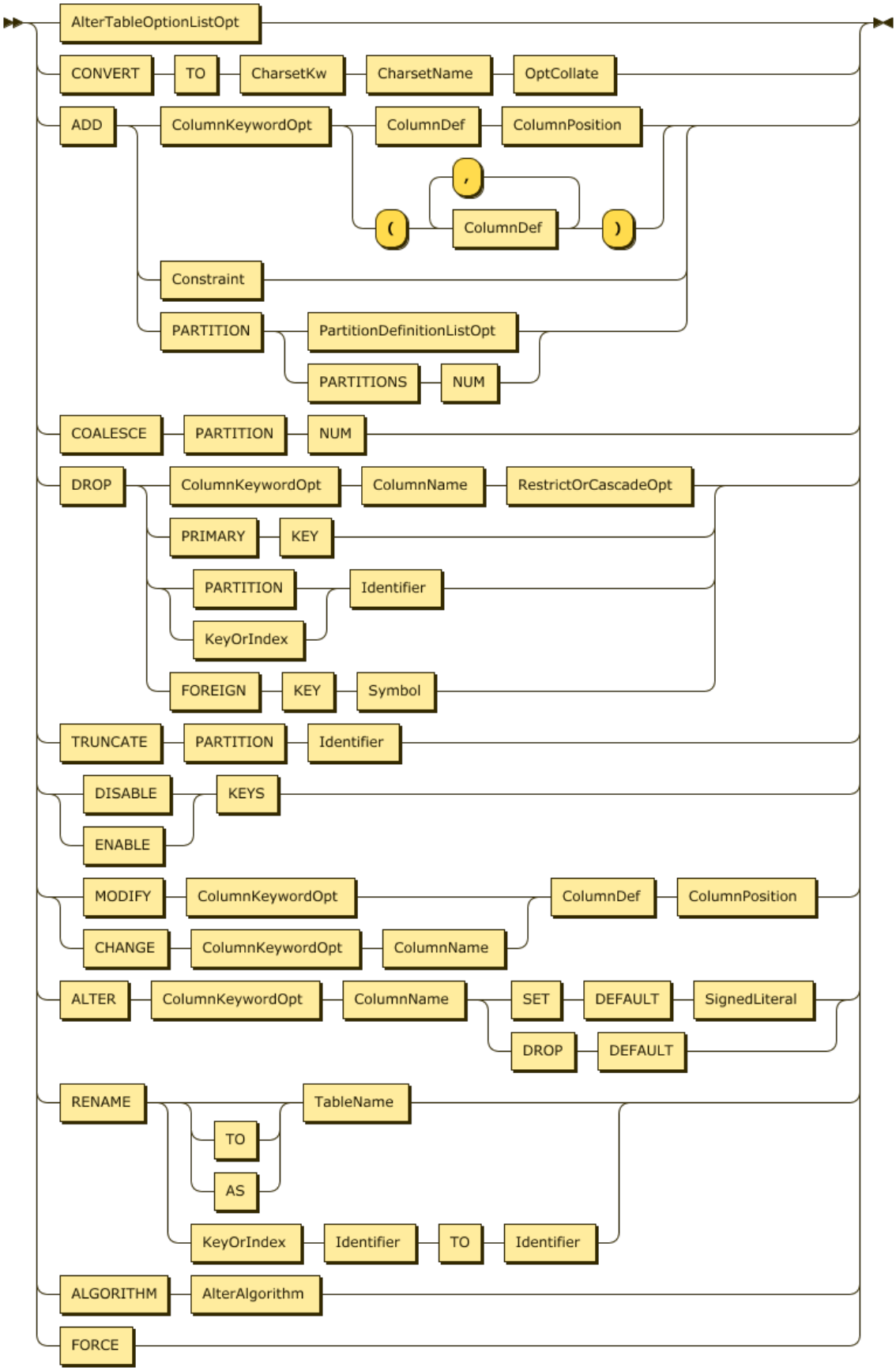

Figure 105: AlterTableSpec

KeyOrIndex:

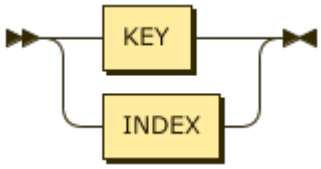

Figure 106: KeyOrIndex

Identifier:

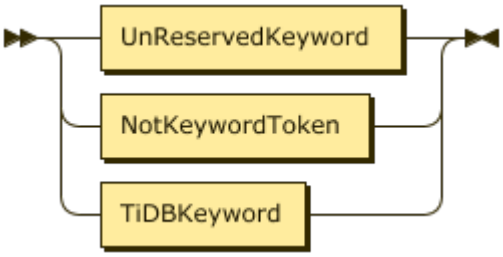

Figure 107: Identifier

4.1.5.29.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.10 sec)

mysql> INSERT INTO t1 (c1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> EXPLAIN SELECT * FROM t1 WHERE c1 = 3;
+--
    ↪ ------------------+---------+------+-------------------------------------
    ↪
| id                | count   | task | operator info
    ↪                                      |
+--
    ↪ ------------------+---------+------+-------------------------------------
    ↪
| TableReader_7     | 10.00   | root | data:Selection_6
    ↪                                       |
|  ─Selection_6     | 10.00   | cop | eq(test.t1.c1, 3)
    ↪                                       |
|    ─TableScan_5   | 10000.00 | cop | table:t1, range:[-inf,+inf], keep
    ↪ order:false, stats:pseudo |
```

```
+--
    ↪ ------------------+----------+------+-------------------------------------------
    ↪
3 rows in set (0.00 sec)

mysql> CREATE INDEX c1 ON t1 (c1);
Query OK, 0 rows affected (0.30 sec)

mysql> EXPLAIN SELECT * FROM t1 WHERE c1 = 3;
+--
    ↪ ----------------+-------+------+-------------------------------------------
    ↪
| id              | count | task | operator info
    ↪                                          |
+--
    ↪ ----------------+-------+------+-------------------------------------------
    ↪
| IndexReader_6   | 10.00 | root | index:IndexScan_5
    ↪                                        |
| └─IndexScan_5   | 10.00 | cop | table:t1, index:c1, range:[3,3], keep
    ↪ order:false, stats:pseudo |
+--
    ↪ ----------------+-------+------+-------------------------------------------
    ↪
2 rows in set (0.00 sec)

mysql> ALTER TABLE t1 DROP INDEX c1;
Query OK, 0 rows affected (0.30 sec)
```

4.1.5.29.3 MySQL compatibility

- Removing the primary key constraint from a column is not supported by default. You can enable the feature by setting the `alter-primary-key` configuration item to `true`. For details, see alter-primary-key.

4.1.5.29.4 See also

- SHOW INDEX
- CREATE INDEX
- ADD INDEX
- RENAME INDEX

4.1.5.30 DROP ROLE

This statement removes a role, that was previously created with `CREATE ROLE`.

4.1.5.30.1 Synopsis

DropRoleStmt:

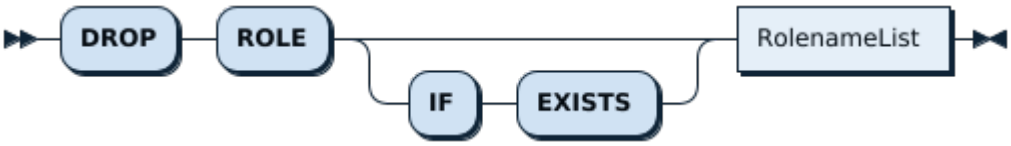

Figure 108: DropRoleStmt

RolenameList:

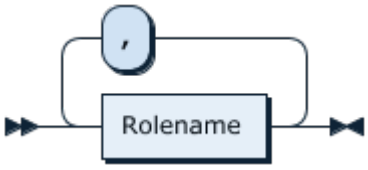

Figure 109: RolenameList

4.1.5.30.2 Examples

Create a new role for the analytics team, and a new user called `jennifer`:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 37
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> CREATE ROLE analyticsteam;
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> GRANT SELECT ON test.* TO analyticsteam;
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE USER jennifer;
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT analyticsteam TO jennifer;
Query OK, 0 rows affected (0.01 sec)
```

Note that by default `jennifer` needs to `SET ROLE analyticsteam` in order to be able to use the privileges associated with the role:

```
$ mysql -ujennifer
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SHOW GRANTS;
+-----------------------------------------+
| Grants for User                         |
+-----------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'    |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+-----------------------------------------+
2 rows in set (0.00 sec)

mysql> SHOW TABLES in test;
ERROR 1044 (42000): Access denied for user 'jennifer'@'%' to database 'test'
mysql> SET ROLE analyticsteam;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS;
+-----------------------------------------+
```

```
| Grants for User                                |
+------------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'           |
| GRANT Select ON test.* TO 'jennifer'@'%'       |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%'    |
+------------------------------------------------+
3 rows in set (0.00 sec)

mysql> SHOW TABLES IN test;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)
```

The statement `SET DEFAULT ROLE` can be used to associated a role to `jennifer` so that she will not have to execute the statement `SET ROLE` in order to assume the privileges associated with the role:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 34
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SET DEFAULT ROLE analyticsteam TO jennifer;
Query OK, 0 rows affected (0.02 sec)
```

```
$ mysql -ujennifer
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 35
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible
```

```
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SHOW GRANTS;
+-----------------------------------------+
| Grants for User                         |
+-----------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'    |
| GRANT Select ON test.* TO 'jennifer'@'%' |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+-----------------------------------------+
3 rows in set (0.00 sec)

mysql> SHOW TABLES IN test;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)
```

Drop the role for the analyticsteam:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 41
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.
```

```
mysql> DROP ROLE analyticsteam;
Query OK, 0 rows affected (0.02 sec)
```

Jennifer no longer has the default role of analyticsteam associated, or can set the role to analyticsteam:

```
$ mysql -ujennifer
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 42
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SHOW GRANTS;
+------------------------------------+
| Grants for User                    |
+------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%' |
+------------------------------------+
1 row in set (0.00 sec)

mysql> SET ROLE analyticsteam;
ERROR 3530 (HY000): `analyticsteam`@`%` is is not granted to jennifer@%
```

4.1.5.30.3 MySQL compatibility

This statement is understood to be fully compatible with roles, which are a feature of MySQL 8.0. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.30.4 See also

- CREATE ROLE
- GRANT <role>
- REVOKE <role>

- SET ROLE
- SET DEFAULT ROLE
- Role-Based Access Control

4.1.5.31 DROP TABLE

This statement drops a table from the currently selected database. An error is returned if the table does not exist, unless the `IF EXISTS` modifier is used.

By design `DROP TABLE` will also drop views, as they share the same namespace as tables.

4.1.5.31.1 Synopsis

DropTableStmt:

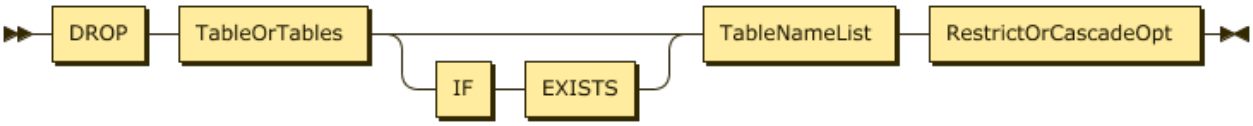

Figure 110: DropTableStmt

TableOrTables:

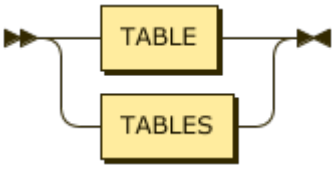

Figure 111: TableOrTables

TableNameList:

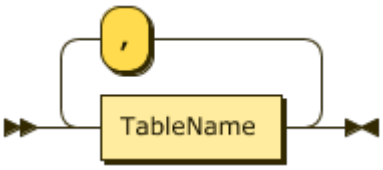

Figure 112: TableNameList

4.1.5.31.2 Examples

```
mysql> CREATE TABLE t1 (a INT);
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> DROP TABLE t1;
Query OK, 0 rows affected (0.22 sec)

mysql> DROP TABLE table_not_exists;
ERROR 1051 (42S02): Unknown table 'test.table_not_exists'
mysql> DROP TABLE IF EXISTS table_not_exists;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE VIEW v1 AS SELECT 1;
Query OK, 0 rows affected (0.10 sec)

mysql> DROP TABLE v1;
Query OK, 0 rows affected (0.23 sec)
```

4.1.5.31.3 MySQL compatibility

- Dropping a table with `IF EXISTS` does not return a warning when attempting to drop a table that does not exist. Issue #7867

4.1.5.31.4 See also

- DROP VIEW
- CREATE TABLE
- SHOW CREATE TABLE
- SHOW TABLES

4.1.5.32 DROP USER

This statement removes a user from the TiDB system database. The optional keyword `IF EXISTS` can be used to silence an error if the user does not exist.

4.1.5.32.1 Synopsis

DropUserStmt:

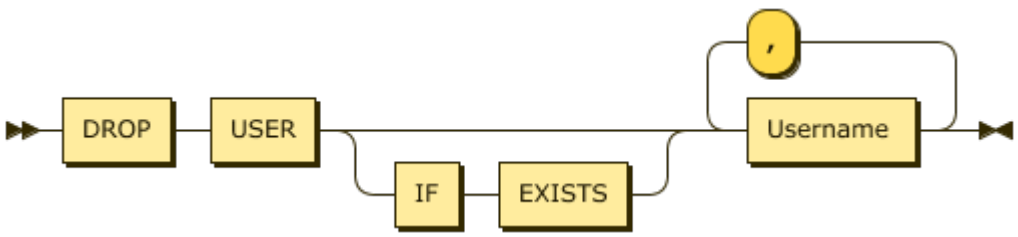

Figure 113: DropUserStmt

Username:

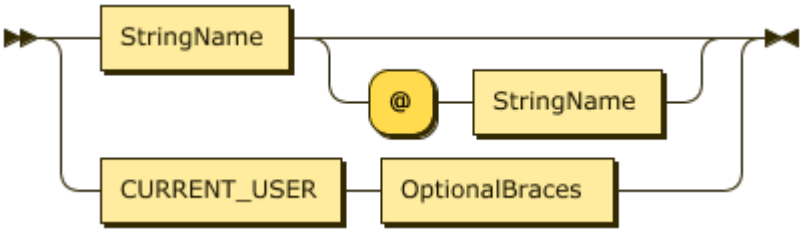

Figure 114: Username

4.1.5.32.2 Examples

```
mysql> DROP USER idontexist;
ERROR 1396 (HY000): Operation DROP USER failed for idontexist@%

mysql> DROP USER IF EXISTS idontexist;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE USER newuser IDENTIFIED BY 'mypassword';
Query OK, 1 row affected (0.02 sec)

mysql> GRANT ALL ON test.* TO 'newuser';
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW GRANTS FOR 'newuser';
+-------------------------------------------------+
| Grants for newuser@%                            |
+-------------------------------------------------+
| GRANT USAGE ON *.* TO 'newuser'@'%'             |
| GRANT ALL PRIVILEGES ON test.* TO 'newuser'@'%' |
+-------------------------------------------------+
2 rows in set (0.00 sec)

mysql> REVOKE ALL ON test.* FROM 'newuser';
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW GRANTS FOR 'newuser';
+-----------------------------------+
| Grants for newuser@%              |
+-----------------------------------+
| GRANT USAGE ON *.* TO 'newuser'@'%' |
+-----------------------------------+
1 row in set (0.00 sec)
```

```
mysql> DROP USER newuser;
Query OK, 0 rows affected (0.14 sec)

mysql> SHOW GRANTS FOR newuser;
ERROR 1141 (42000): There is no such grant defined for user 'newuser' on
    ↪ host '%'
```

4.1.5.32.3 MySQL compatibility

- Dropping a user that does not exist with `IF EXISTS` will not create a warning in TiDB. Issue #10196.

4.1.5.32.4 See also

- CREATE USER
- ALTER USER
- SHOW CREATE USER
- Privilege Management

4.1.5.33 DROP VIEW

This statement drops an view object from the currently selected database. It does not effect any base tables that a view references.

4.1.5.33.1 Synopsis

DropViewStmt:

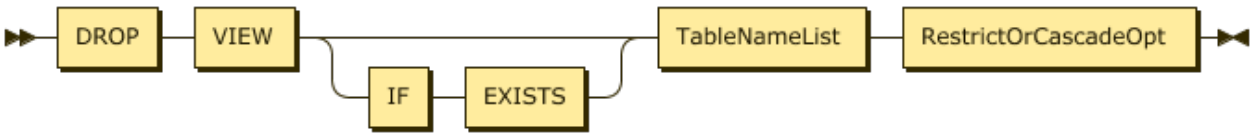

Figure 115: DropViewStmt

TableNameList:

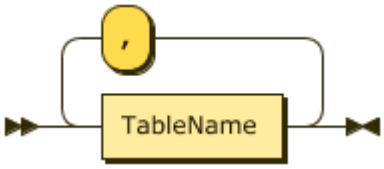

Figure 116: TableNameList

TableName:

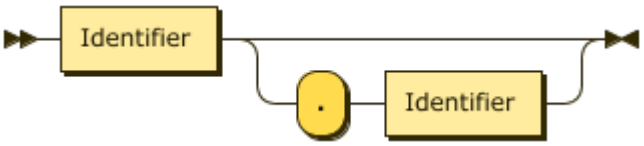

Figure 117: TableName

4.1.5.33.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 (c1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.03 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> CREATE VIEW v1 AS SELECT * FROM t1 WHERE c1 > 2;
Query OK, 0 rows affected (0.11 sec)

mysql> SELECT * FROM t1;
+----+----+
| id | c1 |
+----+----+
|  1 |  1 |
|  2 |  2 |
|  3 |  3 |
|  4 |  4 |
|  5 |  5 |
+----+----+
5 rows in set (0.00 sec)

mysql> SELECT * FROM v1;
+----+----+
| id | c1 |
+----+----+
|  3 |  3 |
|  4 |  4 |
|  5 |  5 |
+----+----+
3 rows in set (0.00 sec)

mysql> DROP VIEW v1;
Query OK, 0 rows affected (0.23 sec)
```

```
mysql> SELECT * FROM t1;
+----+----+
| id | c1 |
+----+----+
|  1 |  1 |
|  2 |  2 |
|  3 |  3 |
|  4 |  4 |
|  5 |  5 |
+----+----+
5 rows in set (0.00 sec)
```

4.1.5.33.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.33.4 See also

- DROP TABLE
- CREATE VIEW

4.1.5.34 EXECUTE

The `EXECUTE` statement provides an SQL interface to server-side prepared statements.

4.1.5.34.1 Synopsis

ExecuteStmt:

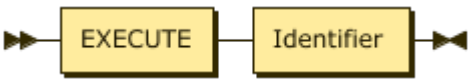

Figure 118: ExecuteStmt

Identifier:

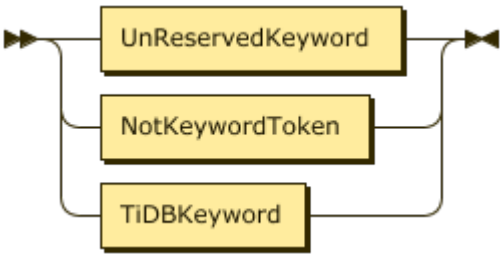

Figure 119: Identifier

4.1.5.34.2 Examples

```
mysql> PREPARE mystmt FROM 'SELECT ? as num FROM DUAL';
Query OK, 0 rows affected (0.00 sec)

mysql> SET @number = 5;
Query OK, 0 rows affected (0.00 sec)

mysql> EXECUTE mystmt USING @number;
+------+
| num |
+------+
| 5   |
+------+
1 row in set (0.00 sec)

mysql> DEALLOCATE PREPARE mystmt;
Query OK, 0 rows affected (0.00 sec)
```

4.1.5.34.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.34.4 See also

- PREPARE
- DEALLOCATE

4.1.5.35 EXPLAIN ANALYZE

The `EXPLAIN ANALYZE` statement works similar to `EXPLAIN`, with the major difference being that it will actually execute the statement. This allows you to compare the estimates used as part of query planning to actual values encountered during execution. If the estimates differ significantly from the actual values, you should consider running `ANALYZE TABLE` on the affected tables.

4.1.5.35.1 Synopsis

ExplainSym:

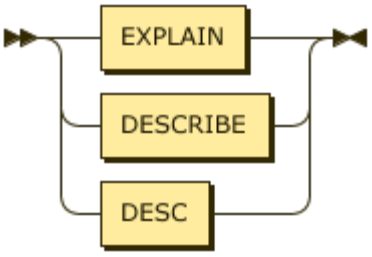

Figure 120: ExplainSym

ExplainStmt:

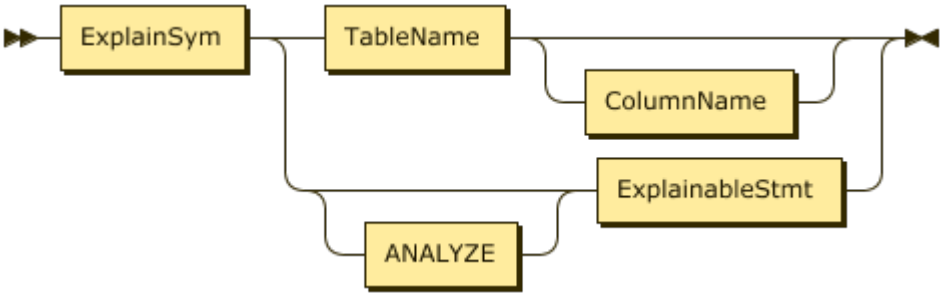

Figure 121: ExplainStmt

ExplainableStmt:

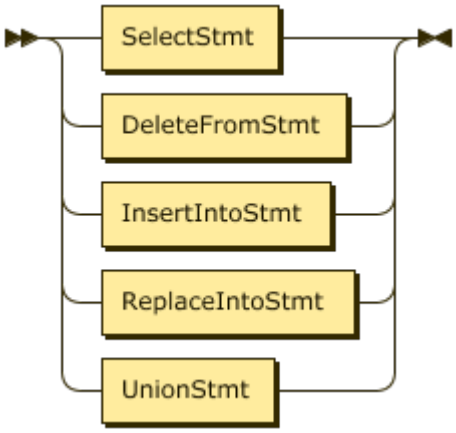

Figure 122: ExplainableStmt

4.1.5.35.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.12 sec)
```

```
mysql> INSERT INTO t1 (c1) VALUES (1), (2), (3);
Query OK, 3 rows affected (0.02 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> EXPLAIN ANALYZE SELECT * FROM t1 WHERE id = 1;
+--
    ↪ ----------+-------+------+-------------------+-------------------------+
    ↪
| id         | count | task | operator info  | execution info          |
+--
    ↪ ----------+-------+------+-------------------+-------------------------+
    ↪
| Point_Get_1 | 1.00 | root | table:t1, handle:1 | time:0ns, loops:0, rows
    ↪ :0 |
+--
    ↪ ----------+-------+------+-------------------+-------------------------+
    ↪
1 row in set (0.01 sec)

mysql> EXPLAIN ANALYZE SELECT * FROM t1;
+--
    ↪ ----------------+----------+------+-----------------------------------------
    ↪
| id              | count    | task | operator info
    ↪                                      | execution info                |
+--
    ↪ ----------------+----------+------+-----------------------------------------
    ↪
| TableReader_5   | 10000.00 | root | data:TableScan_4
    ↪                                     | time:931.759µs, loops:2, rows:3 |
|  ─TableScan_4   | 10000.00 | cop | table:t1, range:[-inf,+inf], keep
    ↪ order:false, stats:pseudo | time:0s, loops:0, rows:3 |
+--
    ↪ ----------------+----------+------+-----------------------------------------
    ↪
2 rows in set (0.00 sec)
```

4.1.5.35.3 MySQL compatibility

`EXPLAIN ANALYZE` is a feature of MySQL 8.0, but both the output format and the potential execution plans in TiDB differ substantially from MySQL.

4.1.5.35.4 See also

- Understanding the Query Execution Plan
- EXPLAIN
- ANALYZE TABLE
- TRACE

4.1.5.36 EXPLAIN

The `EXPLAIN` statement shows the execution plan for a query without executing it. It is complimented by `EXPLAIN ANALYZE` which will execute the query. If the output of `EXPLAIN` does not match the expected result, consider executing `ANALYZE TABLE` on each table in the query.

The statements `DESC` and `DESCRIBE` are aliases of this statement. The alternative usage of `EXPLAIN <tableName>` is documented under `SHOW [FULL] COLUMNS FROM`.

4.1.5.36.1 Synopsis

ExplainSym:

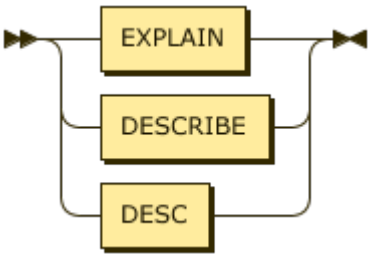

Figure 123: ExplainSym

ExplainStmt:

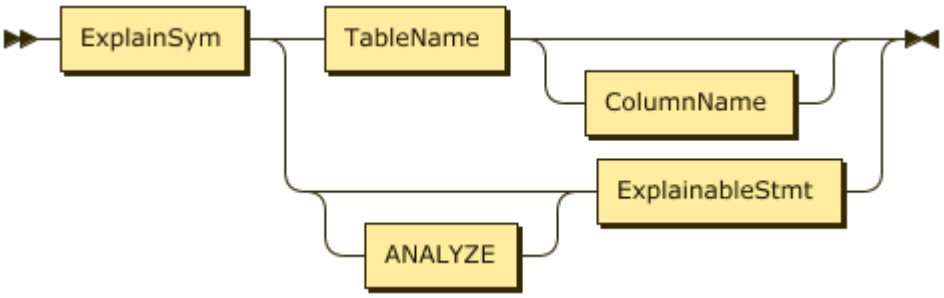

Figure 124: ExplainStmt

ExplainableStmt:

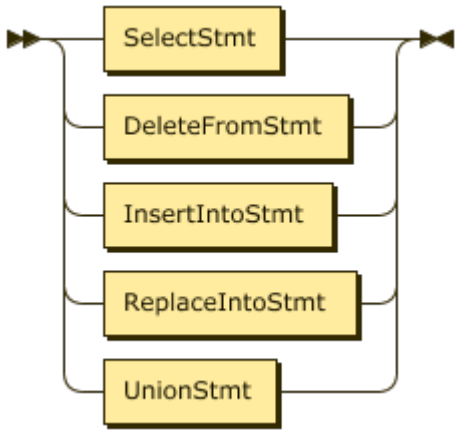

Figure 125: ExplainableStmt

4.1.5.36.2 Examples

```
mysql> EXPLAIN SELECT 1;
+------------------+-------+------+---------------+
| id               | count | task | operator info |
+------------------+-------+------+---------------+
| Projection_3     | 1.00 | root | 1             |
| └─TableDual_4    | 1.00 | root | rows:1        |
+------------------+-------+------+---------------+
2 rows in set (0.00 sec)

mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.10 sec)

mysql> INSERT INTO t1 (c1) VALUES (1), (2), (3);
Query OK, 3 rows affected (0.02 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> EXPLAIN SELECT * FROM t1 WHERE id = 1;
+-------------+-------+------+-------------------+
| id          | count | task | operator info     |
+-------------+-------+------+-------------------+
| Point_Get_1 | 1.00 | root | table:t1, handle:1 |
+-------------+-------+------+-------------------+
1 row in set (0.00 sec)

mysql> DESC SELECT * FROM t1 WHERE id = 1;
+-------------+-------+------+-------------------+
| id          | count | task | operator info     |
+-------------+-------+------+-------------------+
```

```
| Point_Get_1 | 1.00 | root | table:t1, handle:1 |
+-------------+------+------+--------------------+
1 row in set (0.00 sec)

mysql> DESCRIBE SELECT * FROM t1 WHERE id = 1;
+-------------+-------+------+--------------------+
| id          | count | task | operator info      |
+-------------+-------+------+--------------------+
| Point_Get_1 | 1.00  | root | table:t1, handle:1 |
+-------------+-------+------+--------------------+
1 row in set (0.00 sec)

mysql> EXPLAIN INSERT INTO t1 (c1) VALUES (4);
ERROR 1105 (HY000): Unsupported type *core.Insert

mysql> EXPLAIN UPDATE t1 SET c1=5 WHERE c1=3;
+--
   ↪ ------------------+---------+------+-----------------------------------------
   ↪
| id                | count   | task | operator info
   ↪                                        |
+--
   ↪ ------------------+---------+------+-----------------------------------------
   ↪
| TableReader_6     | 10.00   | root | data:Selection_5
   ↪                                       |
|  ─Selection_5     | 10.00   | cop | eq(test.t1.c1, 3)
   ↪                                       |
|    ─TableScan_4   | 10000.00 | cop | table:t1, range:[-inf,+inf], keep
   ↪ order:false, stats:pseudo |
+--
   ↪ ------------------+---------+------+-----------------------------------------
   ↪
3 rows in set (0.00 sec)

mysql> EXPLAIN DELETE FROM t1 WHERE c1=3;
+--
   ↪ ------------------+---------+------+-----------------------------------------
   ↪
| id                | count   | task | operator info
   ↪                                        |
+--
   ↪ ------------------+---------+------+-----------------------------------------
   ↪
| TableReader_6     | 10.00   | root | data:Selection_5
```

```
    ↪                                        |
|  └─Selection_5     | 10.00   | cop | eq(test.t1.c1, 3)
    ↪                                        |
|    └─TableScan_4   | 10000.00 | cop | table:t1, range:[-inf,+inf], keep
    ↪ order:false, stats:pseudo |
+--
    ↪ ------------------+---------+------+-----------------------------------------
    ↪
3 rows in set (0.00 sec)
```

If you do not specify the `FORMAT`, or specify `FORMAT = "row"`, `EXPLAIN` statement will output the results in a tabular format. See Understand the Query Execution Plan for more information.

In addition to the MySQL standard result format, TiDB also supports DotGraph and you need to specify `FORMAT = "dot"` as in the following example:

```
create table t(a bigint, b bigint);
desc format = "dot" select A.a, B.b from t A join t B on A.a > B.b where A.
    ↪ a < 10;

TiDB > desc format = "dot" select A.a, B.b from t A join t B on A.a > B.b
    ↪ where A.a < 10;desc format = "dot" select A.a, B.b from t A join t B
    ↪ on A.a > B.b where A.a < 10;
+--
    ↪ ------------------------------------------------------------------------
    ↪
| dot contents
    ↪
    ↪ |
+--
    ↪ ------------------------------------------------------------------------
    ↪
|
digraph HashRightJoin_7 {
subgraph cluster7{
node [style=filled, color=lightgrey]
color=black
label = "root"
"HashRightJoin_7" -> "TableReader_10"
"HashRightJoin_7" -> "TableReader_12"
}
subgraph cluster9{
node [style=filled, color=lightgrey]
color=black
label = "cop"
```

```
"Selection_9" -> "TableScan_8"
}
subgraph cluster11{
node [style=filled, color=lightgrey]
color=black
label = "cop"
"TableScan_11"
}
"TableReader_10" -> "Selection_9"
"TableReader_12" -> "TableScan_11"
}
 |
+--
    ↪ ------------------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)
```

If the `dot` program (in the `graphviz` package) is installed on your computer, you can generate a PNG file using the following method:

```
dot xx.dot -T png -O

The xx.dot is the result returned by the above statement.
```

If the `dot` program is not installed on your computer, copy the result to this website to get a tree diagram:

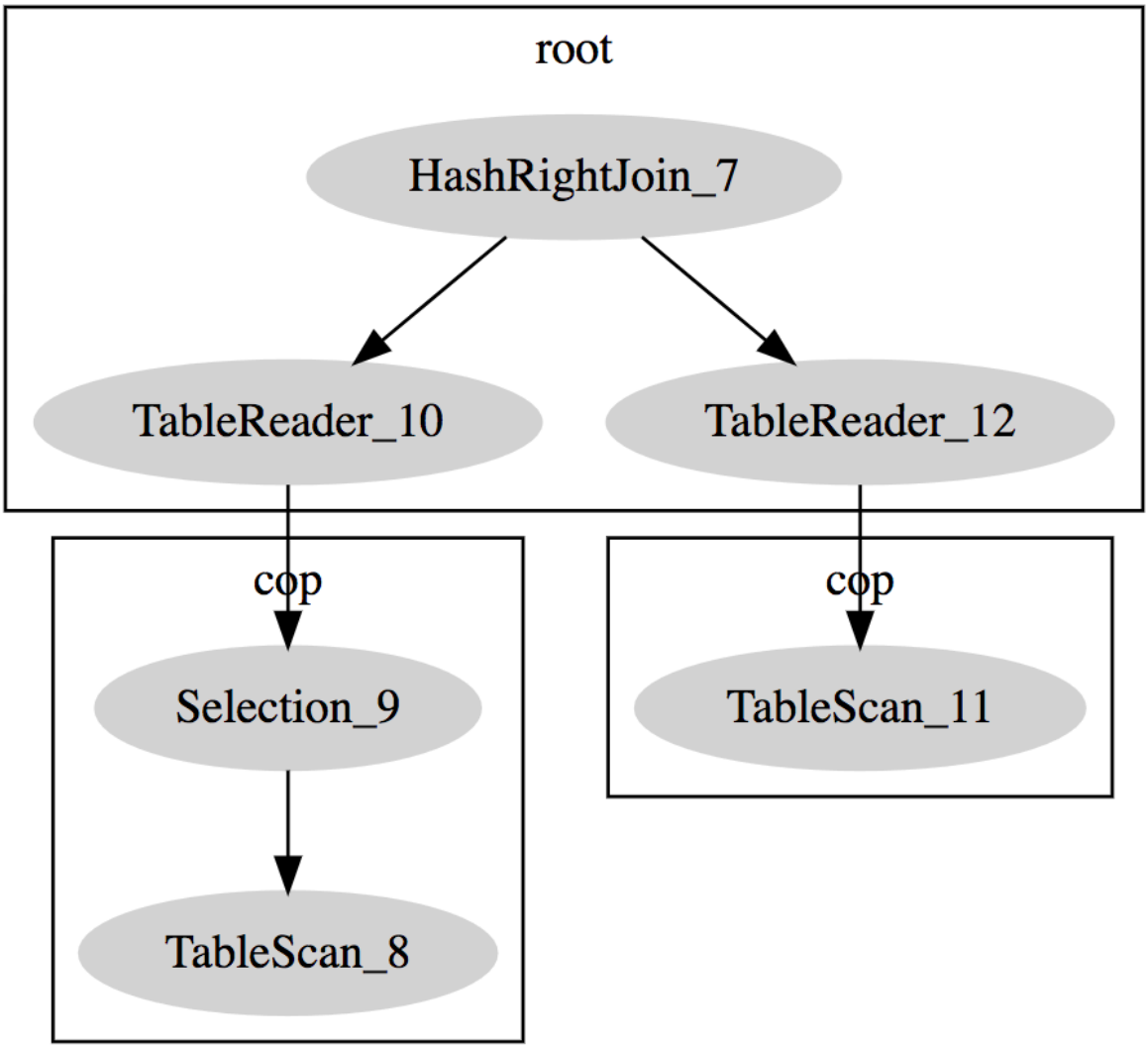

Figure 126: Explain Dot

4.1.5.36.3 MySQL compatibility

- Both the format of `EXPLAIN` and the potential execution plans in TiDB differ substantially from MySQL.
- TiDB does not support the `EXPLAIN FORMAT=JSON` as in MySQL.
- TiDB does not currently support `EXPLAIN` for insert statements.

4.1.5.36.4 See also

- Understanding the Query Execution Plan

- EXPLAIN ANALYZE
- ANALYZE TABLE
- TRACE

4.1.5.37 FLUSH PRIVILEGES

This statement triggers TiDB to reload the in-memory copy of privileges from the privilege tables. You should execute `FLUSH PRIVILEGES` after making manual edits to tables such as `mysql.user`. Executing this statement is not required after using privilege statements such as `GRANT` or `REVOKE`.

4.1.5.37.1 Synopsis

FlushStmt:

Figure 127: FlushStmt

NoWriteToBinLogAliasOpt:

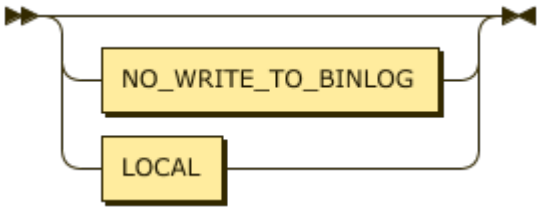

Figure 128: NoWriteToBinLogAliasOpt

FlushOption:

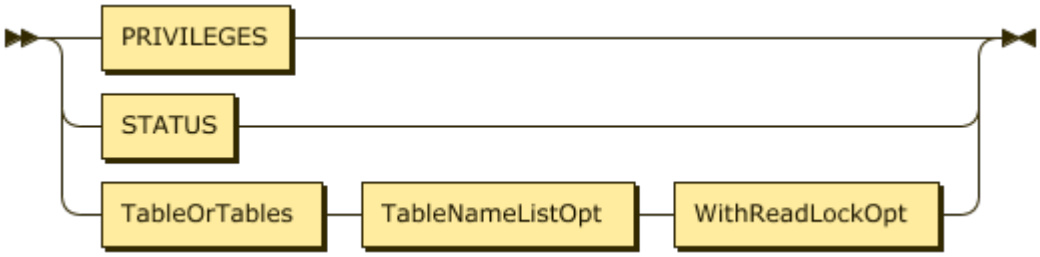

Figure 129: FlushOption

4.1.5.37.2 Examples

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

4.1.5.37.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.37.4 See also

- Privilege Management

4.1.5.38 FLUSH STATUS

This statement is included for compatibility with MySQL. It has no effect on TiDB, which uses Prometheus and Grafana for centralized metrics collection instead of `SHOW STATUS`.

4.1.5.38.1 Synopsis

FlushStmt:

Figure 130: FlushStmt

NoWriteToBinLogAliasOpt:

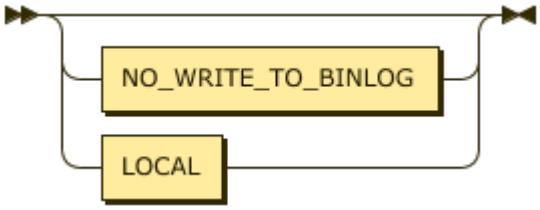

Figure 131: NoWriteToBinLogAliasOpt

FlushOption:

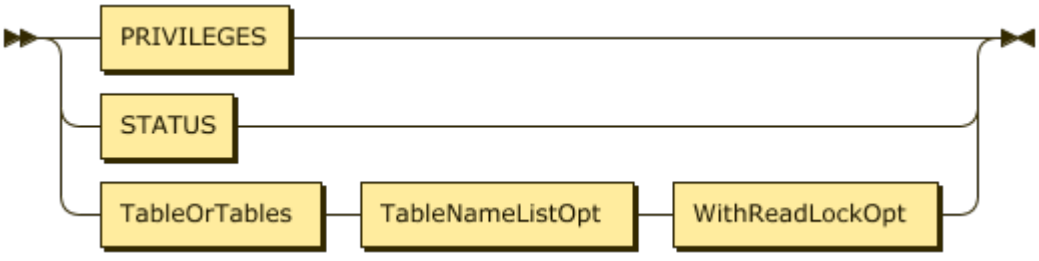

Figure 132: FlushOption

4.1.5.38.2 Examples

```
mysql> show status;
+--------------------+--------------------------------------+
| Variable_name    | Value                                |
+--------------------+--------------------------------------+
| Ssl_cipher_list  |                                      |
| server_id        | 93e2e07d-6bb4-4a1b-90b7-e035fae154fe |
| ddl_schema_version | 141                                |
| Ssl_verify_mode  | 0                                    |
| Ssl_version      |                                      |
| Ssl_cipher       |                                      |
+--------------------+--------------------------------------+
6 rows in set (0.01 sec)

mysql> show global status;
+--------------------+--------------------------------------+
| Variable_name    | Value                                |
+--------------------+--------------------------------------+
| Ssl_cipher       |                                      |
| Ssl_cipher_list  |                                      |
| Ssl_verify_mode  | 0                                    |
| Ssl_version      |                                      |
| server_id        | 93e2e07d-6bb4-4a1b-90b7-e035fae154fe |
| ddl_schema_version | 141                                |
+--------------------+--------------------------------------+
6 rows in set (0.00 sec)

mysql> flush status;
Query OK, 0 rows affected (0.00 sec)

mysql> show status;
+--------------------+--------------------------------------+
| Variable_name    | Value                                |
+--------------------+--------------------------------------+
| Ssl_cipher       |                                      |
| Ssl_cipher_list  |                                      |
| Ssl_verify_mode  | 0                                    |
| Ssl_version      |                                      |
| ddl_schema_version | 141                                |
| server_id        | 93e2e07d-6bb4-4a1b-90b7-e035fae154fe |
+--------------------+--------------------------------------+
6 rows in set (0.00 sec)
```

4.1.5.38.3 MySQL compatibility

- This statement is included only for compatibility with MySQL.

4.1.5.38.4 See also

- SHOW [GLOBAL|SESSION] STATUS

4.1.5.39 FLUSH TABLES

This statement is included for compatibility with MySQL. It has no effective usage in TiDB.

4.1.5.39.1 Synopsis

FlushStmt:

Figure 133: FlushStmt

NoWriteToBinLogAliasOpt:

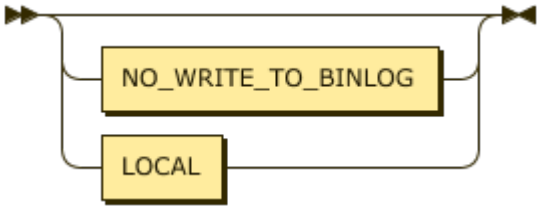

Figure 134: NoWriteToBinLogAliasOpt

FlushOption:

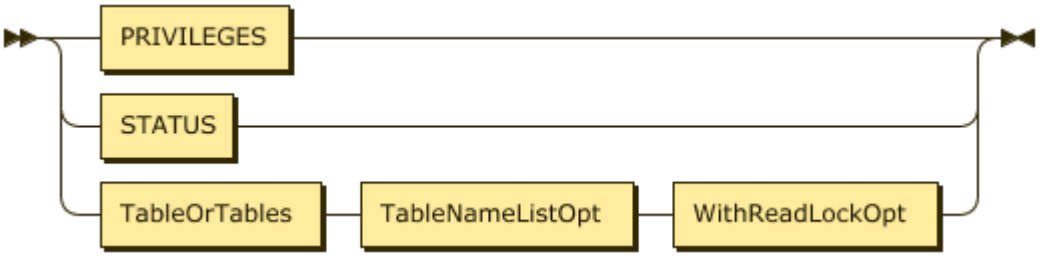

Figure 135: FlushOption

TableOrTables:

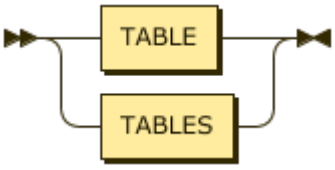

Figure 136: TableOrTables

TableNameListOpt:

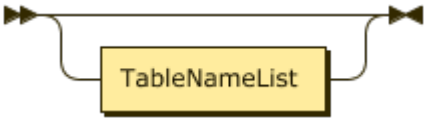

Figure 137: TableNameListOpt

WithReadLockOpt:

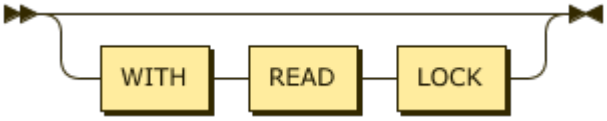

Figure 138: WithReadLockOpt

4.1.5.39.2 Examples

```
mysql> FLUSH TABLES;
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH TABLES WITH READ LOCK;
ERROR 1105 (HY000): FLUSH TABLES WITH READ LOCK is not supported. Please
    ↪ use @@tidb_snapshot
```

4.1.5.39.3 MySQL compatibility

- TiDB does not have a concept of table cache as in MySQL. Thus, `FLUSH TABLES` is parsed but ignored in TiDB for compatibility.
- The statement `FLUSH TABLES WITH READ LOCK` produces an error, as TiDB does not currently support locking tables. It is recommended to use Historical reads for this purpose instead.

4.1.5.39.4 See also

- Read historical data

4.1.5.40 `GRANT <privileges>`

This statement allocates privileges to a pre-existing user in TiDB. The privilege system in TiDB follows MySQL, where credentials are assigned based on a database/table pattern.

4.1.5.40.1 Synopsis

GrantStmt:

Figure 139: GrantStmt

PrivElemList:

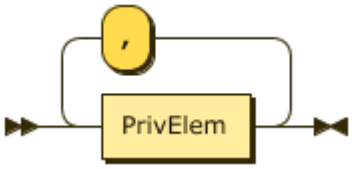

Figure 140: PrivElemList

PrivElem:

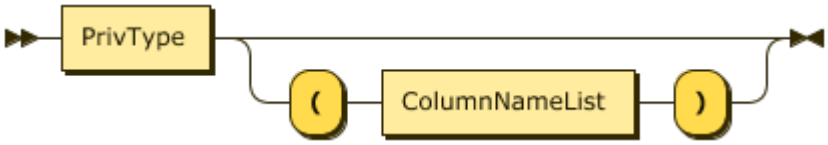

Figure 141: PrivElem

PrivType:

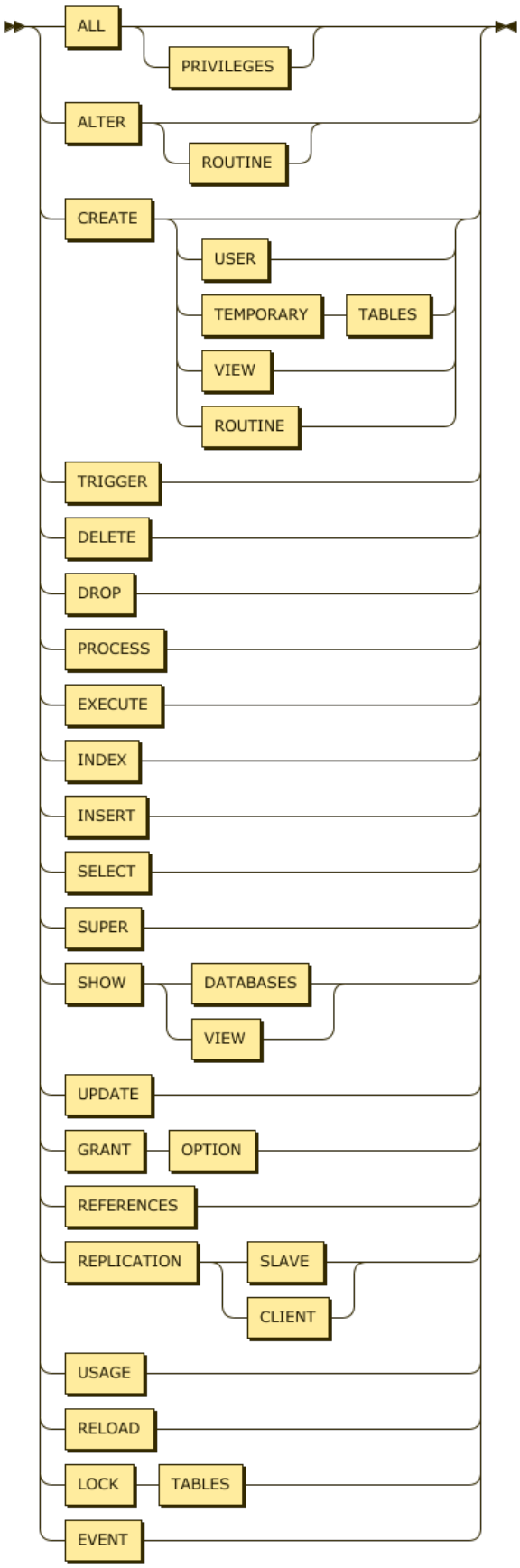

Figure 142: PrivType

ObjectType:

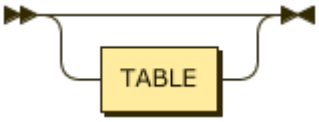

Figure 143: ObjectType

PrivLevel:

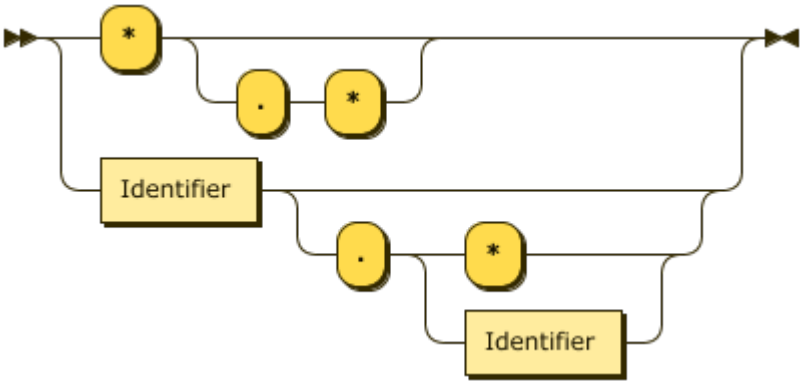

Figure 144: PrivLevel

UserSpecList:

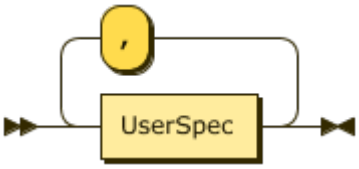

Figure 145: UserSpecList

4.1.5.40.2 Examples

```
mysql> CREATE USER newuser IDENTIFIED BY 'mypassword';
Query OK, 1 row affected (0.02 sec)

mysql> GRANT ALL ON test.* TO 'newuser';
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW GRANTS FOR 'newuser';
+-------------------------------------------------+
| Grants for newuser@%                            |
+-------------------------------------------------+
```

```
| GRANT USAGE ON *.* TO 'newuser'@'%'       |
| GRANT ALL PRIVILEGES ON test.* TO 'newuser'@'%' |
+-------------------------------------------------+
2 rows in set (0.00 sec)
```

4.1.5.40.3 MySQL compatibility

- Similar to MySQL, the `USAGE` privilege denotes the ability to log into a TiDB server.
- Column level privileges are not currently supported.
- Similar to MySQL, when the `NO_AUTO_CREATE_USER` sql mode is not present, the `GRANT` statement will automatically create a new user with an empty password when a user does not exist. Removing this sql-mode (it is enabled by default) presents a security risk.

4.1.5.40.4 See also

- GRANT <role>
- REVOKE <privileges>
- SHOW GRANTS
- Privilege Management

4.1.5.41 `GRANT <role>`

Assigns a previously created role to an existing user. The user can use then use the statement `SET ROLE <rolename>` to assume the privileges of the role, or `SET ROLE ALL` to assume all roles that have been assigned.

4.1.5.41.1 Synopsis

GrantRoleStmt:

Figure 146: GrantRoleStmt

RolenameList:

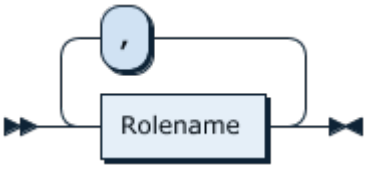

Figure 147: RolenameList

UsernameList:

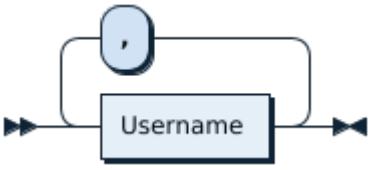

Figure 148: UsernameList

4.1.5.41.2 Examples

Create a new role for the analytics team, and a new user called `jennifer`:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 37
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> CREATE ROLE analyticsteam;
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT SELECT ON test.* TO analyticsteam;
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE USER jennifer;
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT analyticsteam TO jennifer;
Query OK, 0 rows affected (0.01 sec)
```

Note that by default `jennifer` needs to `SET ROLE analyticsteam` in order to be able to use the privileges associated with the role:

```
$ mysql -ujennifer
Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Your MySQL connection id is 32
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SHOW GRANTS;
+----------------------------------------------+
| Grants for User                              |
+----------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'         |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+----------------------------------------------+
2 rows in set (0.00 sec)

mysql> SHOW TABLES in test;
ERROR 1044 (42000): Access denied for user 'jennifer'@'%' to database 'test'
mysql> SET ROLE analyticsteam;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS;
+----------------------------------------------+
| Grants for User                              |
+----------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'         |
| GRANT Select ON test.* TO 'jennifer'@'%'     |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+----------------------------------------------+
3 rows in set (0.00 sec)

mysql> SHOW TABLES IN test;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)
```

The statement `SET DEFAULT ROLE` can be used to associated a role to `jennifer` so that she will not have to execute the statement `SET ROLE` in order to assume the privileges associated with the role:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 34
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SET DEFAULT ROLE analyticsteam TO jennifer;
Query OK, 0 rows affected (0.02 sec)
```

```
$ mysql -ujennifer
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 35
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SHOW GRANTS;
+-------------------------------------------+
| Grants for User                           |
+-------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'  |
```

```
| GRANT Select ON test.* TO 'jennifer'@'%' |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+---------------------------------------------+
3 rows in set (0.00 sec)

mysql> SHOW TABLES IN test;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)
```

4.1.5.41.3 MySQL compatibility

This statement is understood to be fully compatible with roles, which are a feature of MySQL 8.0. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.41.4 See also

- GRANT <privileges>
- CREATE ROLE
- DROP ROLE
- REVOKE <role>
- SET ROLE
- SET DEFAULT ROLE
- Role-Based Access Control

4.1.5.42 INSERT

This statement inserts new rows into a table.

4.1.5.42.1 Synopsis

InsertIntoStmt:

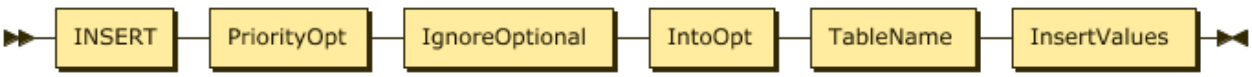

Figure 149: InsertIntoStmt

PriorityOpt:

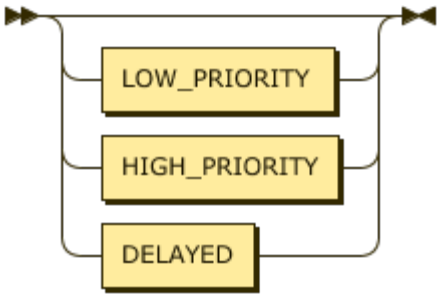

Figure 150: PriorityOpt

IgnoreOptional:

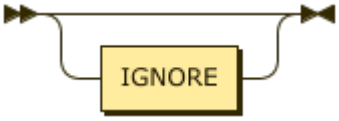

Figure 151: IgnoreOptional

IntoOpt:

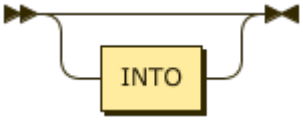

Figure 152: IntoOpt

TableName:

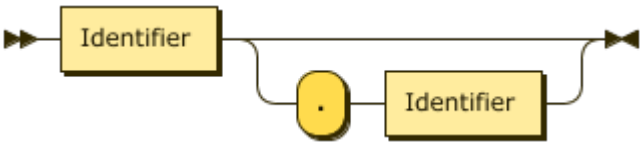

Figure 153: TableName

InsertValues:

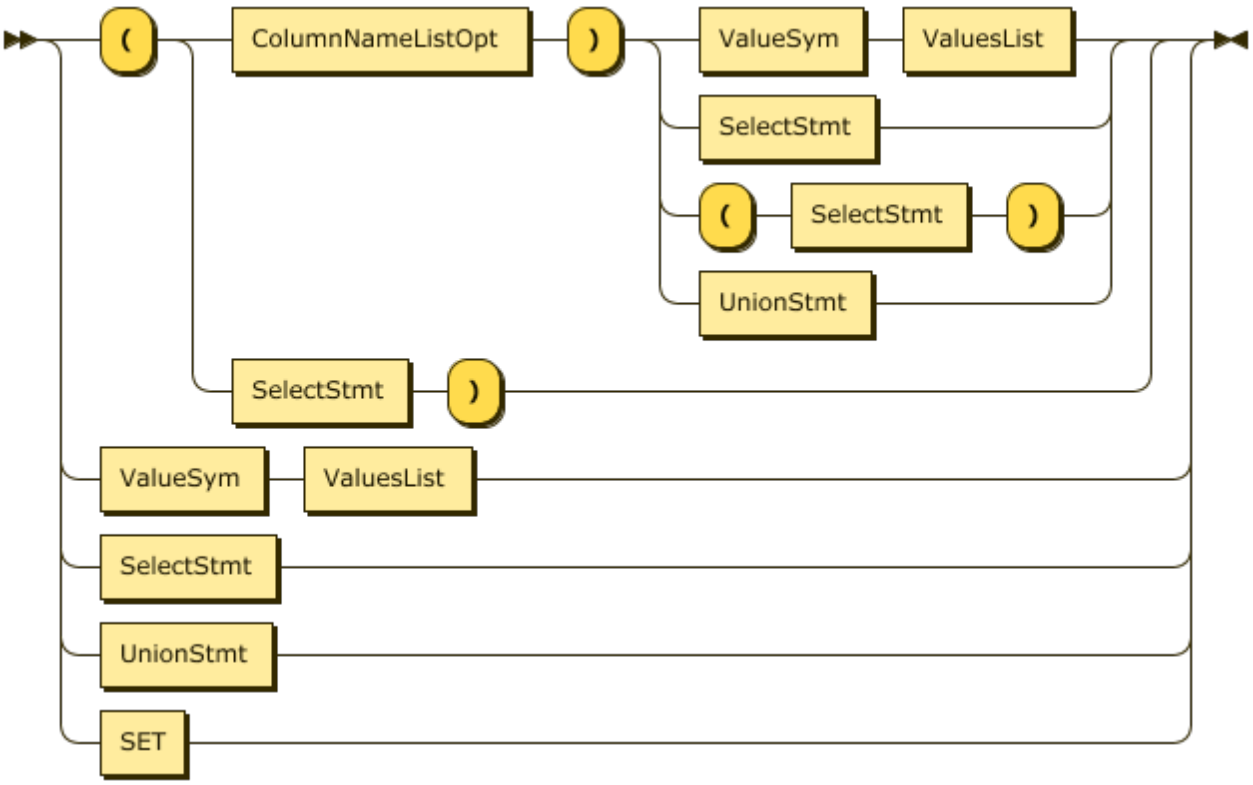

Figure 154: InsertValues

4.1.5.42.2 Examples

```
mysql> CREATE TABLE t1 (a int);
Query OK, 0 rows affected (0.11 sec)

mysql> CREATE TABLE t2 LIKE t1;
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 VALUES (1);
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO t1 (a) VALUES (1);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO t2 SELECT * FROM t1;
Query OK, 2 rows affected (0.01 sec)
Records: 2 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM t1;
+------+
| a    |
```

```
+------+
|    1 |
|    1 |
+------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM t2;
+------+
| a    |
+------+
|    1 |
|    1 |
+------+
2 rows in set (0.00 sec)

mysql> INSERT INTO t2 VALUES (2),(3),(4);
Query OK, 3 rows affected (0.02 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM t2;
+------+
| a    |
+------+
|    1 |
|    1 |
|    2 |
|    3 |
|    4 |
+------+
5 rows in set (0.00 sec)
```

4.1.5.42.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.42.4 See also

- DELETE
- SELECT
- UPDATE
- REPLACE

4.1.5.43 KILL TIDB

The statement `KILL TIDB` is used to terminate connections in TiDB.

4.1.5.43.1 Synopsis

KillStmt:

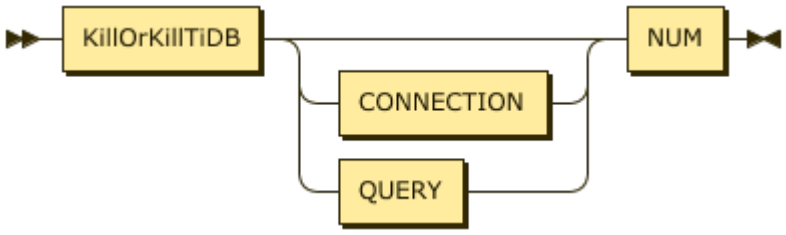

Figure 155: KillStmt

4.1.5.43.2 Examples

```
mysql> SHOW PROCESSLIST;
+--
    ↪ ----+------+-----------+------+---------+------+-------+------------------+
    ↪
| Id   | User | Host      | db   | Command | Time | State | Info             |
+--
    ↪ ----+------+-----------+------+---------+------+-------+------------------+
    ↪
|    1 | root | 127.0.0.1 | test | Query | 0 | 2     | SHOW PROCESSLIST |
|    2 | root | 127.0.0.1 |   | Sleep  |    4 | 2     |                  |
+--
    ↪ ----+------+-----------+------+---------+------+-------+------------------+
    ↪
2 rows in set (0.00 sec)

mysql> KILL TIDB 2;
Query OK, 0 rows affected (0.00 sec)
```

4.1.5.43.3 MySQL compatibility

- By design, this statement is not compatible with MySQL by default. This helps prevent against a case of a connection being terminated on the wrong TiDB server, because it is common to place multiple TiDB servers behind a load balancer.
- The `KILL TIDB` statement is a TiDB extension. If you are certain that the session you are attempting to kill is on the same TiDB server, set `compatible-kill-query = true` in your configuration file.

4.1.5.43.4 See also

- SHOW [FULL] PROCESSLIST

4.1.5.44 LOAD DATA

The `LOAD DATA` statement batch loads data into a TiDB table.

4.1.5.44.1 Synopsis

LoadDataStmt:

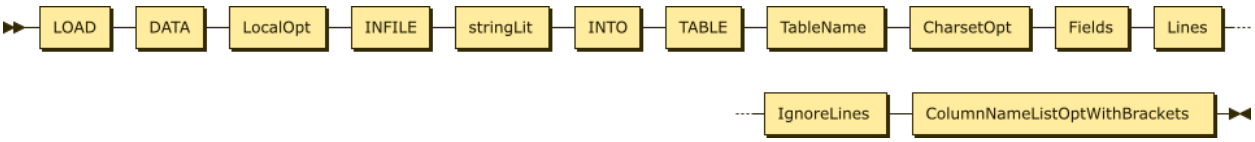

Figure 156: LoadDataStmt

4.1.5.44.2 Parameters

`Fields` and `Lines`

You can specify how to process the data format by configuring the `Fields` and `Lines` parameters.

- `FIELDS TERMINATED BY`: Specifies the separating character of each data.
- `FIELDS ENCLOSED BY`: Specifies the enclosing character of each data.
- `LINES TERMINATED BY`: Specifies the line terminator, if you want to end a line with a certain character.

Take the following data format as an example:

```
"bob","20","street 1"\r\n
"alice","33","street 1"\r\n
```

If you want to extract `bob`, `20`, and `street 1`, specify the separating character as `','`, and the enclosing character as `'\"'`:

```
FIELDS TERMINATED BY ',' ENCLOSED BY '\"' LINES TERMINATED BY '\r\n'
```

If you do not specify the parameters above, the imported data is processed in the following way by default:

```
FIELDS TERMINATED BY '\t' ENCLOSED BY ''
LINES TERMINATED BY '\n'
```

`IGNORE number LINES`

You can ignore the first `number` lines of a file by configuring the `IGNORE number LINES` parameter. For example, if you configure `IGNORE 1 LINES`, the first line of a file is ignored.

4.1.5.44.3 Examples

```
CREATE TABLE trips (
    -> trip_id bigint NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> duration integer not null,
    -> start_date datetime,
    -> end_date datetime,
    -> start_station_number integer,
    -> start_station varchar(255),
    -> end_station_number integer,
    -> end_station varchar(255),
    -> bike_number varchar(255),
    -> member_type varchar(255)
    -> );
```

```
Query OK, 0 rows affected (0.14 sec)
```

The following example imports data using `LOAD DATA`. Comma is specified as the separating character. The double quotation marks that enclose the data is ignored. The first line of the file is ignored.

If you see the error message `ERROR 1148 (42000): the used command is not allowed with this TiDB version`, refer to ERROR 1148 (42000): the used command is not allowed with this TiDB version.

```
LOAD DATA LOCAL INFILE '/mnt/evo970/data-sets/bikeshare-data/2017Q4-
    ↪ capitalbikeshare-tripdata.csv' INTO TABLE trips FIELDS TERMINATED BY
    ↪ ',' ENCLOSED BY '\"' LINES TERMINATED BY '\r\n' IGNORE 1 LINES (
    ↪ duration, start_date, end_date, start_station_number, start_station,
    ↪ end_station_number, end_station, bike_number, member_type);
```

```
Query OK, 815264 rows affected (39.63 sec)
Records: 815264 Deleted: 0 Skipped: 0 Warnings: 0
```

4.1.5.44.4 MySQL compatibility

- TiDB will by default commit every 20 000 rows. This behavior is similar to MySQL NDB Cluster, but not the default configuration with the InnoDB storage engine.

Note:

- Committing through splitting a transaction is at the expense of breaking the atomicity and isolation of the transaction. When performing this operation, you must ensure that there are **no other** ongoing operations on the table. When an error occurs, **manual intervention is required to check the consistency and integrity of the data**. Therefore, it is not recommended to use `LOAD DATA` on any tables which are actively being read from or written to.
- No matter how many rows are committed in a transaction, `LOAD DATA` is not rolled back by the `ROLLBACK` statement in an explicit transaction.
- The `LOAD DATA` statement is always executed in optimistic transaction mode, regardless of the TiDB transaction mode configuration.

4.1.5.44.5 See also

- INSERT
- Import Example Database
- TiDB Optimistic Transaction Model
- TiDB Pessimistic Transaction Mode

4.1.5.45 LOAD STATS

The `LOAD STATS` statement is used to load the statistics into TiDB.

4.1.5.45.1 Synopsis

LoadStatsStmt:

Figure 157: LoadStatsStmt

4.1.5.45.2 Examples

You can access the address `http://${tidb-server-ip}:${tidb-server-status-port}/stats/dump/${db_name}/${table_name}` to download the TiDB instance's statistics.

You can also use `LOAD STATS ${stats_path}` to load the specific statistics file.

The `${stats_path}` can be an absolute path or a relative path. If you use a relative path, the corresponding file is found starting from the path where `tidb-server` is started. Here is an example:

```
LOAD STATS '/tmp/stats.json';
```

```
Query OK, 0 rows affected (0.00 sec)
```

4.1.5.45.3 MySQL compatibility

This statement is a TiDB extension to MySQL syntax.

4.1.5.45.4 See also

- Statistics

4.1.5.46 MODIFY COLUMN

The `ALTER TABLE.. MODIFY COLUMN` statement modifies a column on an existing table. The modification can include changing the data type and attributes. To rename at the same time, use the `CHANGE COLUMN` statement instead.

4.1.5.46.1 Synopsis

AlterTableStmt:

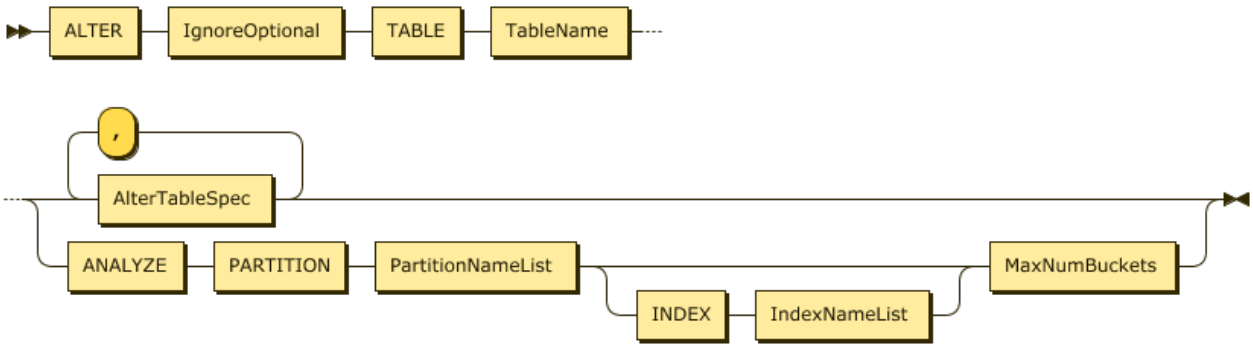

Figure 158: AlterTableStmt

AlterTableSpec:

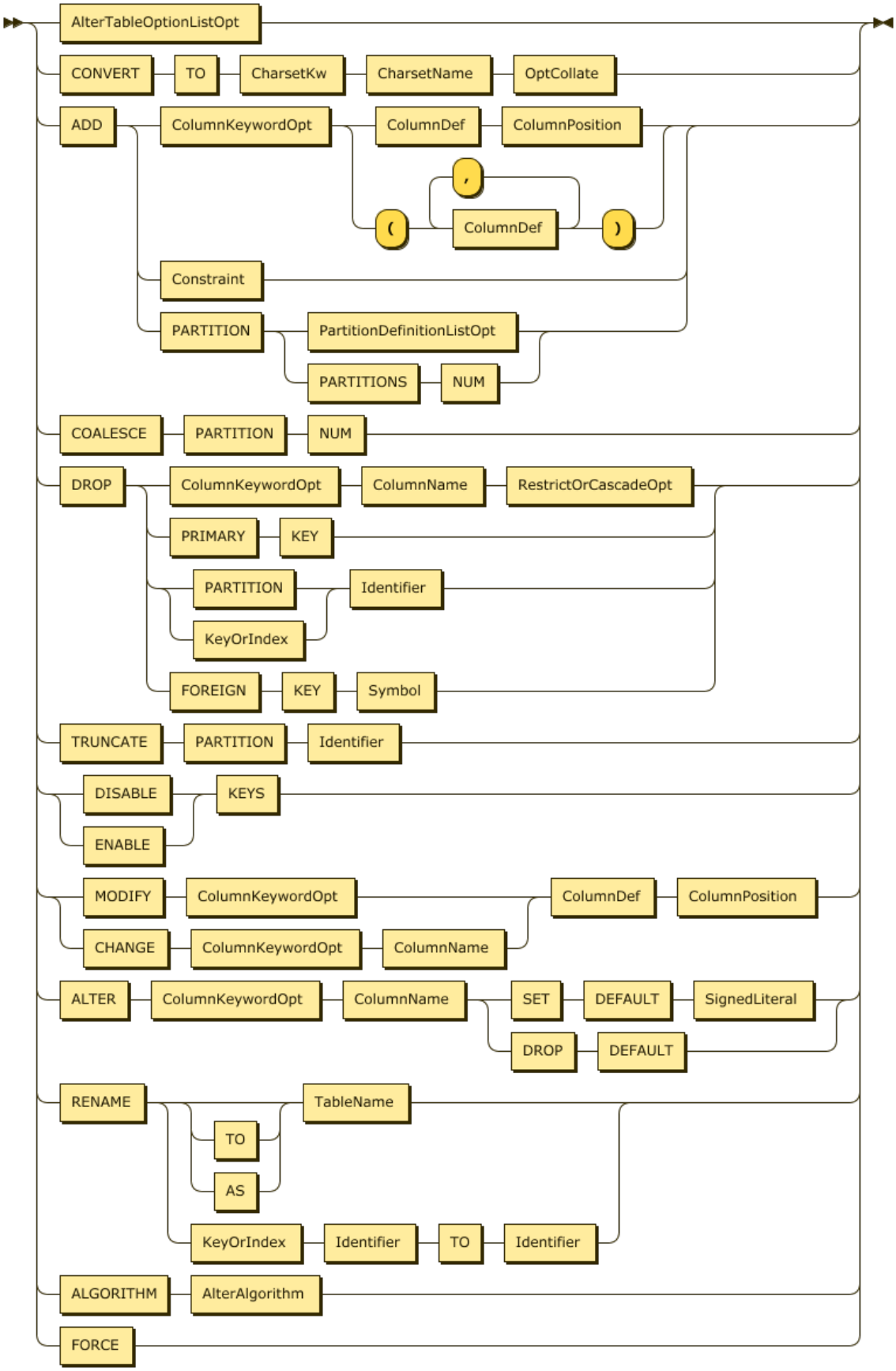

Figure 159: AlterTableSpec

ColumnKeywordOpt:

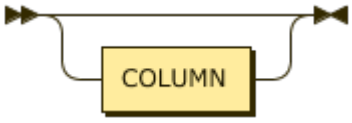

Figure 160: ColumnKeywordOpt

ColumnDef:

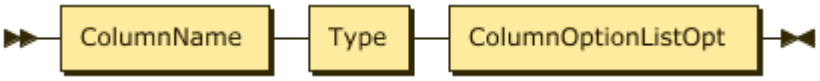

Figure 161: ColumnDef

ColumnPosition:

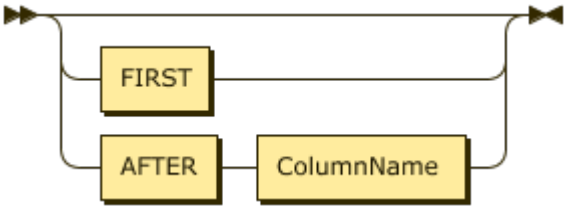

Figure 162: ColumnPosition

4.1.5.46.2 Examples

```
mysql> CREATE TABLE t1 (id int not null primary key AUTO_INCREMENT, col1
    ↪ INT);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 (col1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE t1 MODIFY col1 BIGINT;
Query OK, 0 rows affected (0.09 sec)

mysql> SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `col1` bigint(20) DEFAULT NULL,
```

```
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin AUTO_INCREMENT
    ↪ =30001
1 row in set (0.00 sec)

mysql> ALTER TABLE t1 MODIFY col1 INT;
ERROR 1105 (HY000): unsupported modify column length 11 is less than origin
    ↪  20
mysql> ALTER TABLE t1 MODIFY col1 BLOB;
ERROR 1105 (HY000): unsupported modify column type 252 not match origin 8
mysql> ALTER TABLE t1 MODIFY col1 BIGINT, MODIFY id BIGINT NOT NULL;
ERROR 1105 (HY000): can't run multi schema change
```

4.1.5.46.3 MySQL compatibility

- Making multiple changes in a single `ALTER TABLE` statement is not currently supported.
- Only certain types of data type changes are supported. For example, an `INTEGER` to `BIGINT` is supported, but the reverse is not possible. Changing from an integer to a string format or blob is not supported.
- Modifying precision of the `DECIMAL` data type is not supported.

4.1.5.46.4 See also

- CREATE TABLE
- SHOW CREATE TABLE
- ADD COLUMN
- DROP COLUMN
- CHANGE COLUMN

4.1.5.47 PREPARE

The `PREPARE` statement provides an SQL interface to server-side prepared statements.

4.1.5.47.1 Synopsis

PreparedStmt:

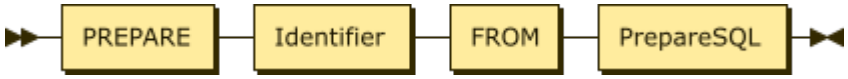

Figure 163: PreparedStmt

4.1.5.47.2 Examples

```
mysql> PREPARE mystmt FROM 'SELECT ? as num FROM DUAL';
Query OK, 0 rows affected (0.00 sec)

mysql> SET @number = 5;
Query OK, 0 rows affected (0.00 sec)

mysql> EXECUTE mystmt USING @number;
+------+
| num |
+------+
| 5   |
+------+
1 row in set (0.00 sec)

mysql> DEALLOCATE PREPARE mystmt;
Query OK, 0 rows affected (0.00 sec)
```

4.1.5.47.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.47.4 See also

- EXECUTE
- DEALLOCATE

4.1.5.48 RECOVER TABLE

`RECOVER TABLE` is used to recover a deleted table and the data on it within the GC (Garbage Collection) life time after the `DROP TABLE` statement is executed.

4.1.5.48.1 Syntax

```
RECOVER TABLE table_name
```

```
RECOVER TABLE BY JOB ddl_job_id
```

Note:

- If a table is deleted and the GC lifetime is out, the table cannot be recovered with `RECOVER TABLE`. Execution of `RECOVER TABLE` in this scenario returns an error like: `snapshot is older than GC safe point 2019-07-10 13:45:57 +0800 CST`.
- If the TiDB version is 3.0.0 or later, it is not recommended for you to use `RECOVER TABLE` when TiDB Binlog is used.
- `RECOVER TABLE` is supported in the Binlog version 3.0.1, so you can use `RECOVER TABLE` in the following three situations:
  - Binlog version is 3.0.1 or later.
  - TiDB 3.0 is used both in the upstream cluster and the downstream cluster.
  - The GC life time of the secondary cluster must be longer than that of the primary cluster. However, as latency occurs during data replication between upstream and downstream databases, data recovery might fail in the downstream.

Troubleshoot errors during TiDB Binlog replication

When you use `RECOVER TABLE` in the upstream TiDB during TiDB Binlog replication, TiDB Binlog might be interrupted in the following three situations:

- The downstream database does not support the `RECOVER TABLE` statement. An error instance: `check the manual that corresponds to your MySQL server version for the right syntax to use near 'RECOVER TABLE table_name'`.
- The GC life time is not consistent between the upstream database and the downstream database. An error instance: `snapshot is older than GC safe point 2019-07-10 13:45:57 +0800 CST`.
- Latency occurs during replication between upstream and downstream databases. An error instance: `snapshot is older than GC safe point 2019-07-10 13:45:57 +0800 CST`.

For the above three situations, you can resume data replication from TiDB Binlog with a full import of the deleted table.

4.1.5.48.2 Examples

- Recover the deleted table according to the table name.

```
DROP TABLE t;
```

```
RECOVER TABLE t;
```

This method searches the recent DDL job history and locates the first DDL operation of the `DROP TABLE` type, and then recovers the deleted table with the name identical to the one table name specified in the `RECOVER TABLE` statement.

- Recover the deleted table according to the table's `DDL JOB ID` used.

  Suppose that you had deleted the table `t` and created another `t`, and again you deleted the newly created `t`. Then, if you want to recover the `t` deleted in the first place, you must use the method that specifies the `DDL JOB ID`.

```
DROP TABLE t;
```

```
ADMIN SHOW DDL JOBS 1;
```

The second statement above is used to search for the table's `DDL JOB ID` to delete `t`. In the following example, the ID is 53.

```
+--------+---------+-----------+-----------+-------------+-----------+----------
   ↪
| JOB_ID | DB_NAME | TABLE_NAME | JOB_TYPE | SCHEMA_STATE | SCHEMA_ID |
   ↪  TABLE_ID | ROW_COUNT | START_TIME          | STATE |
+--------+---------+-----------+-----------+-------------+-----------+----------
   ↪
| 53     | test    |           | drop table | none       | 1         | 41
   ↪      | 0         | 2019-07-10 13:23:18.277 +0800 CST | synced |
+--------+---------+-----------+-----------+-------------+-----------+----------
   ↪
```

```
RECOVER TABLE BY JOB 53;
```

This method recovers the deleted table via the `DDL JOB ID`. If the corresponding DDL job is not of the `DROP TABLE` type, an error occurs.

4.1.5.48.3 Implementation principle

When deleting a table, TiDB only deletes the table metadata, and writes the table data (row data and index data) to be deleted to the `mysql.gc_delete_range` table. The GC Worker in the TiDB background periodically removes from the `mysql.gc_delete_range` table the keys that exceed the GC life time.

Therefore, to recover a table, you only need to recover the table metadata and delete the corresponding row record in the `mysql.gc_delete_range` table before the GC Worker deletes the table data. You can use a snapshot read of TiDB to recover the table metadata. Refer to Read Historical Data for details.

Table recovery is done by TiDB obtaining the table metadata through snapshot read, and then going through the process of table creation similar to `CREATE TABLE`. Therefore, `RECOVER TABLE` itself is, in essence, a kind of DDL operation.

4.1.5.48.4 MySQL compatibility

This statement is a TiDB extension to MySQL syntax.

4.1.5.49 RENAME INDEX

The statement `ALTER TABLE .. RENAME INDEX` renames an existing index to a new name. This operation is instant in TiDB, and requires only a meta data change.

4.1.5.49.1 Synopsis

AlterTableStmt:

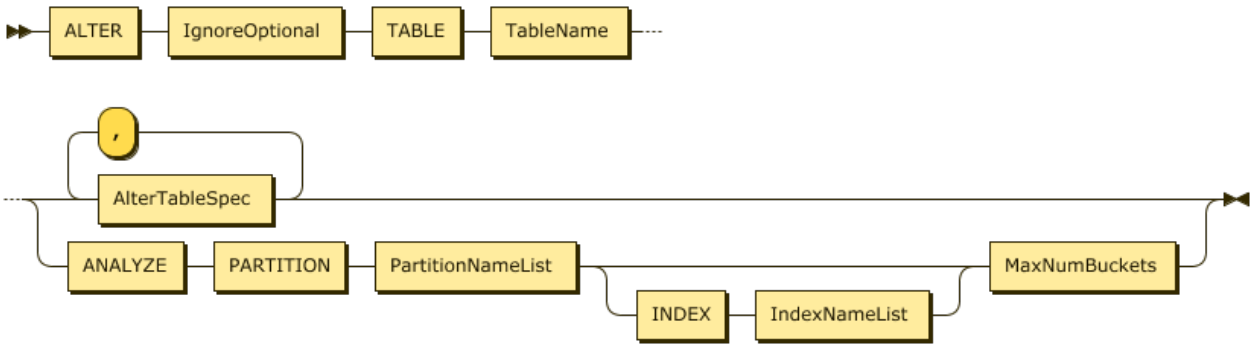

Figure 164: AlterTableStmt

KeyOrIndex:

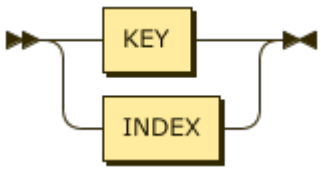

Figure 165: KeyOrIndex

4.1.5.49.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL, INDEX col1 (c1));
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `c1` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `col1` (`c1`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
1 row in set (0.00 sec)

mysql> ALTER TABLE t1 RENAME INDEX col1 TO c1;
Query OK, 0 rows affected (0.09 sec)

mysql> SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `c1` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `c1` (`c1`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
1 row in set (0.00 sec)
```

4.1.5.49.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.49.4 See also

- SHOW CREATE TABLE
- CREATE INDEX
- DROP INDEX
- SHOW INDEX

4.1.5.50 RENAME TABLE

This statement renames an existing table to a new name.

4.1.5.50.1 Synopsis

RenameTableStmt:

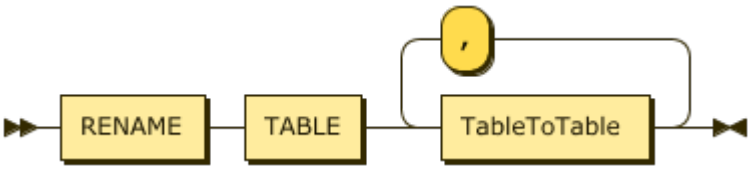

Figure 166: RenameTableStmt

TableToTable:

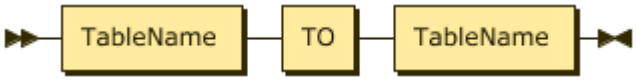

Figure 167: TableToTable

4.1.5.50.2 Examples

```
mysql> CREATE TABLE t1 (a int);
Query OK, 0 rows affected (0.12 sec)

mysql> SHOW TABLES;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)

mysql> RENAME TABLE t1 TO t2;
Query OK, 0 rows affected (0.08 sec)

mysql> SHOW TABLES;
+----------------+
| Tables_in_test |
+----------------+
| t2             |
+----------------+
1 row in set (0.00 sec)
```

4.1.5.50.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.50.4 See also

- CREATE TABLE
- SHOW TABLES
- ALTER TABLE

4.1.5.51 REPLACE

The `REPLACE` statement is semantically a combined `DELETE`+`INSERT` statement. It can be used to simplify application code.

4.1.5.51.1 Synopsis

ReplaceIntoStmt:

Figure 168: ReplaceIntoStmt

PriorityOpt:

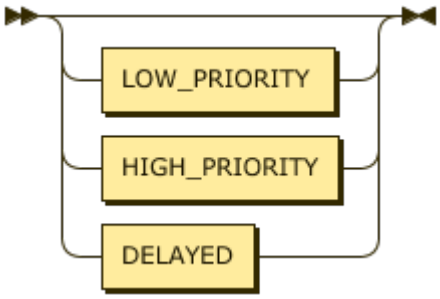

Figure 169: PriorityOpt

IntoOpt:

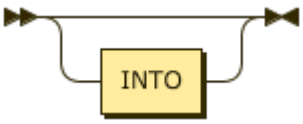

Figure 170: IntoOpt

TableName:

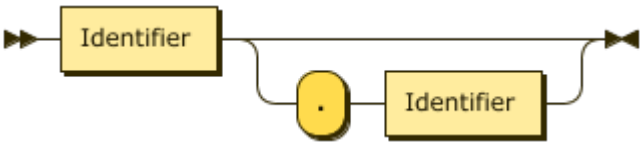

Figure 171: TableName

InsertValues:

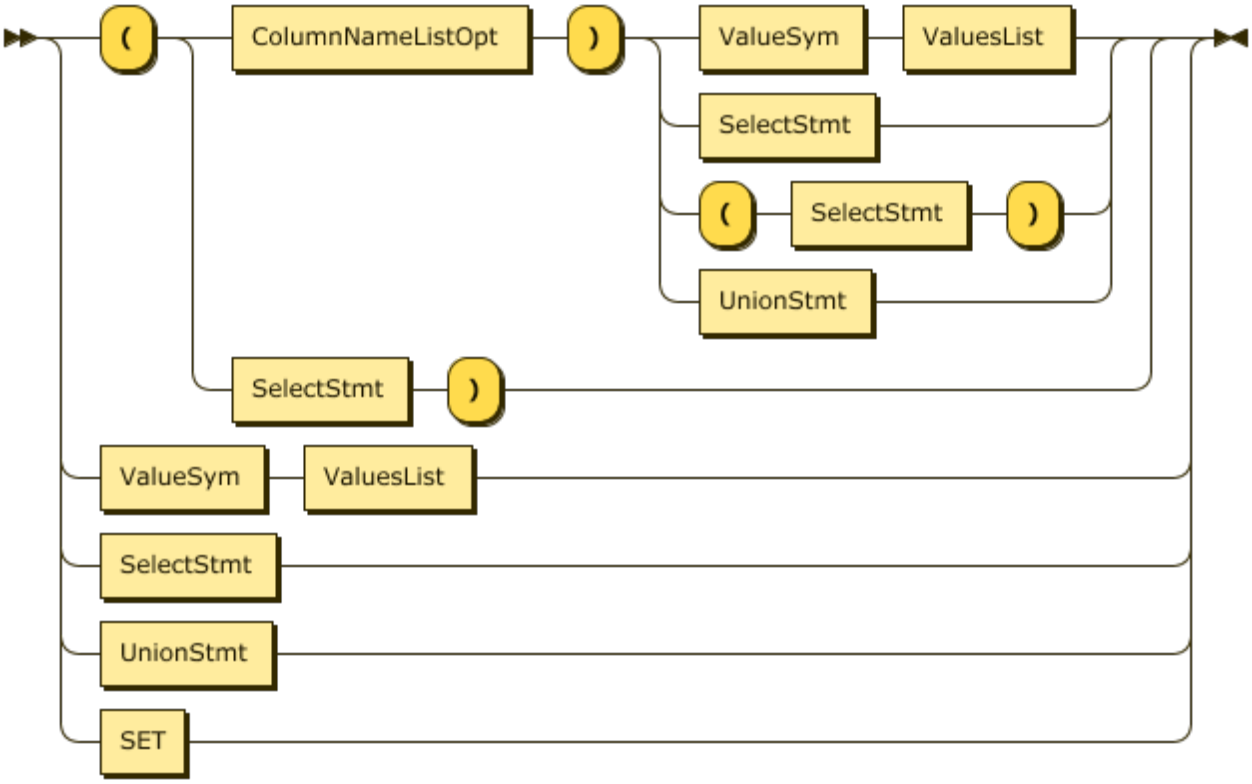

Figure 172: InsertValues

4.1.5.51.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.12 sec)

mysql> INSERT INTO t1 (c1) VALUES (1), (2), (3);
Query OK, 3 rows affected (0.02 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM t1;
+----+----+
```

```
| id | c1 |
+----+----+
|  1 |  1 |
|  2 |  2 |
|  3 |  3 |
+----+----+
3 rows in set (0.00 sec)

mysql> REPLACE INTO t1 (id, c1) VALUES(3, 99);
Query OK, 2 rows affected (0.01 sec)

mysql> SELECT * FROM t1;
+----+----+
| id | c1 |
+----+----+
|  1 |  1 |
|  2 |  2 |
|  3 | 99 |
+----+----+
3 rows in set (0.00 sec)
```

4.1.5.51.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.51.4 See also

- DELETE
- INSERT
- SELECT
- UPDATE

4.1.5.52 `REVOKE <privileges>`

This statement removes privileges from an existing user.

4.1.5.52.1 Synopsis

GrantStmt:

Figure 173: GrantStmt

PrivElemList:

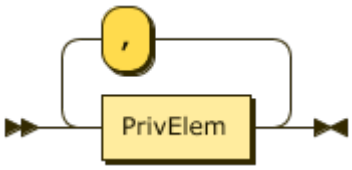

Figure 174: PrivElemList

PrivElem:

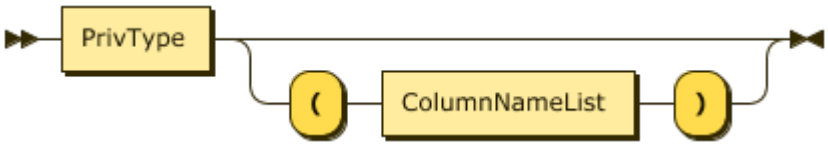

Figure 175: PrivElem

PrivType:

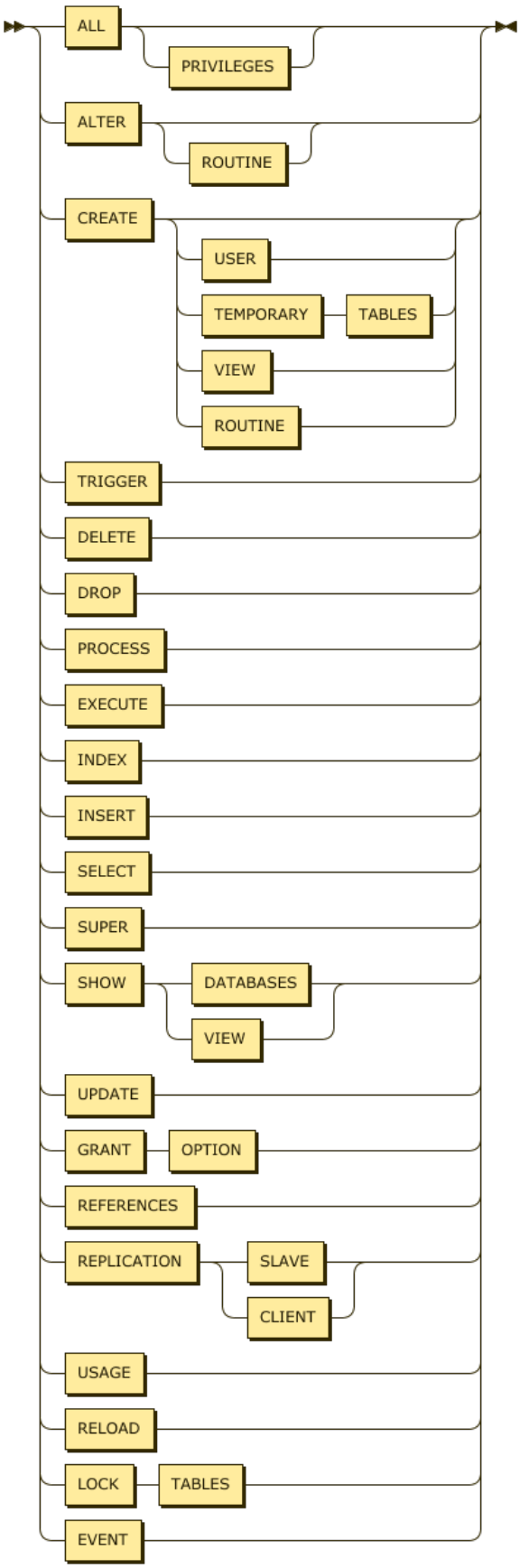

Figure 176: PrivType

ObjectType:

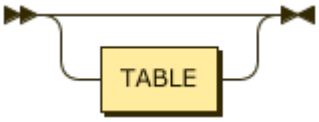

Figure 177: ObjectType

PrivLevel:

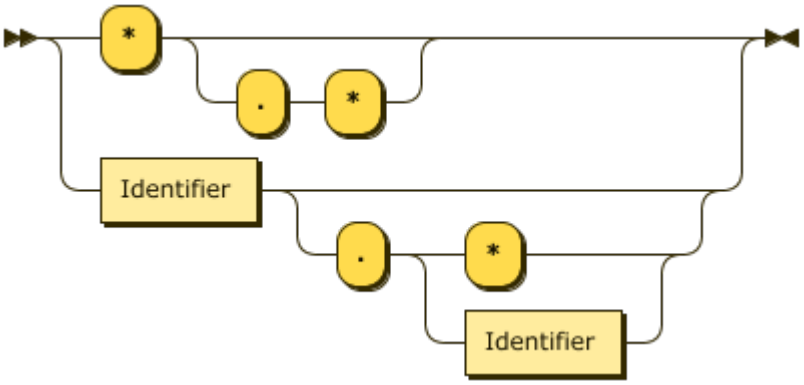

Figure 178: PrivLevel

UserSpecList:

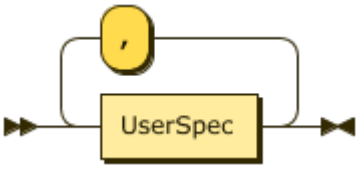

Figure 179: UserSpecList

4.1.5.52.2 Examples

```
mysql> CREATE USER newuser IDENTIFIED BY 'mypassword';
Query OK, 1 row affected (0.02 sec)

mysql> GRANT ALL ON test.* TO 'newuser';
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW GRANTS FOR 'newuser';
+-------------------------------------------------+
| Grants for newuser@%                            |
+-------------------------------------------------+
```

```
| GRANT USAGE ON *.* TO 'newuser'@'%'       |
| GRANT ALL PRIVILEGES ON test.* TO 'newuser'@'%' |
+-------------------------------------------------+
2 rows in set (0.00 sec)

mysql> REVOKE ALL ON test.* FROM 'newuser';
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW GRANTS FOR 'newuser';
+-----------------------------------+
| Grants for newuser@%              |
+-----------------------------------+
| GRANT USAGE ON *.* TO 'newuser'@'%' |
+-----------------------------------+
1 row in set (0.00 sec)

mysql> DROP USER newuser;
Query OK, 0 rows affected (0.14 sec)

mysql> SHOW GRANTS FOR newuser;
ERROR 1141 (42000): There is no such grant defined for user 'newuser' on
    ↪ host '%'
```

4.1.5.52.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.52.4 See also

- GRANT <privileges>
- SHOW GRANTS
- Privilege Management

4.1.5.53 REVOKE <role>

This statement removes a previously assigned role from a specified user (or list of users).

4.1.5.53.1 Synopsis

RevokeRoleStmt:

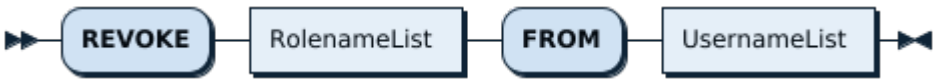

Figure 180: RevokeRoleStmt

RolenameList:

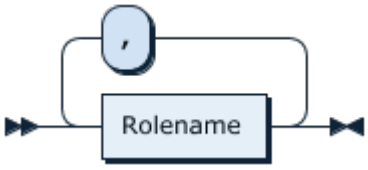

Figure 181: RolenameList

UsernameList:

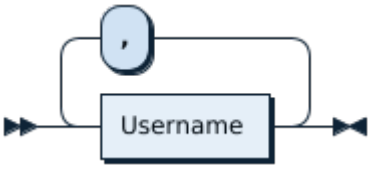

Figure 182: UsernameList

4.1.5.53.2 Examples

Create a new role for the analytics team, and a new user called `jennifer`:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 37
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> CREATE ROLE analyticsteam;
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT SELECT ON test.* TO analyticsteam;
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE USER jennifer;
```

```
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT analyticsteam TO jennifer;
Query OK, 0 rows affected (0.01 sec)
```

Note that by default `jennifer` needs to `SET ROLE analyticsteam` in order to be able to use the privileges associated with the role:

```
$ mysql -ujennifer
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SHOW GRANTS;
+-----------------------------------------------+
| Grants for User                               |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'          |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+-----------------------------------------------+
2 rows in set (0.00 sec)

mysql> SHOW TABLES in test;
ERROR 1044 (42000): Access denied for user 'jennifer'@'%' to database 'test'
mysql> SET ROLE analyticsteam;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS;
+-----------------------------------------------+
| Grants for User                               |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'          |
| GRANT Select ON test.* TO 'jennifer'@'%' |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
```

```
+-----------------------------------------+
3 rows in set (0.00 sec)

mysql> SHOW TABLES IN test;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)
```

The statement `SET DEFAULT ROLE` can be used to associated a role to `jennifer` so that she will not have to execute the statement `SET ROLE` in order to assume the privileges associated with the role:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 34
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SET DEFAULT ROLE analyticsteam TO jennifer;
Query OK, 0 rows affected (0.02 sec)
```

```
$ mysql -ujennifer
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 35
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
```

```
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SHOW GRANTS;
+-------------------------------------------+
| Grants for User                           |
+-------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'      |
| GRANT Select ON test.* TO 'jennifer'@'%'  |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+-------------------------------------------+
3 rows in set (0.00 sec)

mysql> SHOW TABLES IN test;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)
```

Revoke the role of analyticsteam from `jennifer`:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 38
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> REVOKE analyticsteam FROM jennifer;
Query OK, 0 rows affected (0.01 sec)
```

```
$ mysql -ujennifer
```

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 39
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SHOW GRANTS;
+------------------------------------+
| Grants for User                    |
+------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%' |
+------------------------------------+
1 row in set (0.00 sec)
```

4.1.5.53.3 MySQL compatibility

This statement is understood to be fully compatible with roles, which are a feature of MySQL 8.0. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.53.4 See also

- CREATE ROLE
- DROP ROLE
- GRANT <role>
- SET ROLE
- SET DEFAULT ROLE
- Role-Based Access Control

4.1.5.54 ROLLBACK

This statement reverts all changes in the current transaction inside of TIDB. It is the opposite of a `COMMIT` statement.

4.1.5.54.1 Synopsis

Statement:

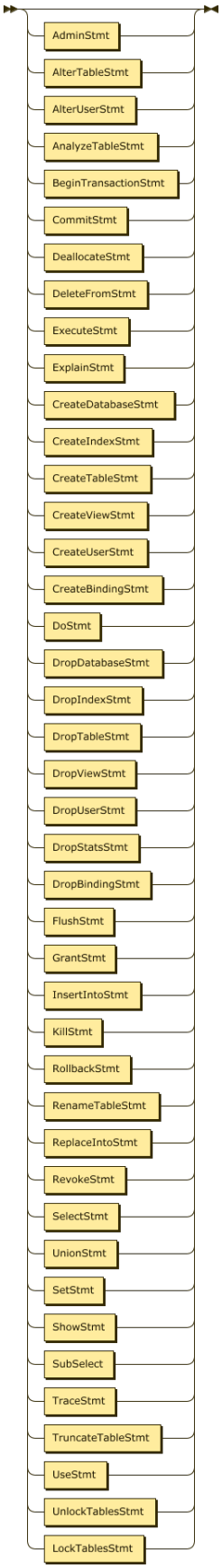

Figure 183: Statement

4.1.5.54.2 Examples

```
mysql> CREATE TABLE t1 (a int NOT NULL PRIMARY KEY);
Query OK, 0 rows affected (0.12 sec)

mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t1 VALUES (1);
Query OK, 1 row affected (0.00 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM t1;
Empty set (0.01 sec)
```

4.1.5.54.3 MySQL compatibility

TiDB does not support the syntax `ROLLBACK TO SAVEPOINT`. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.54.4 See also

- COMMIT
- BEGIN
- START TRANSACTION

4.1.5.55 SELECT

The `SELECT` statement is used to read data from TiDB.

4.1.5.55.1 Synopsis

SelectStmt:

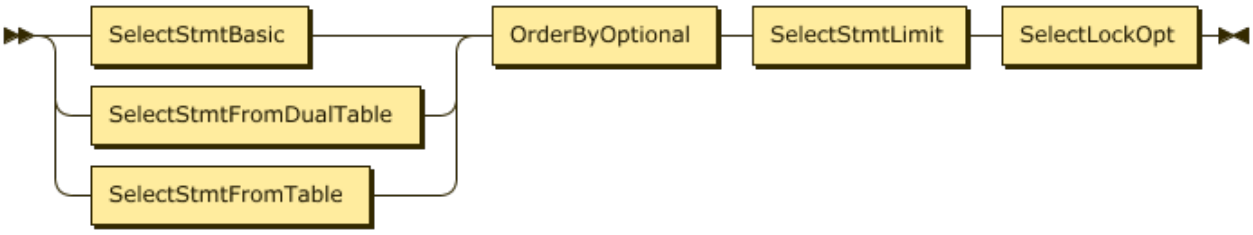

Figure 184: SelectStmt

FromDual:

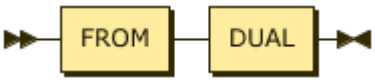

Figure 185: FromDual

WhereClauseOptional:

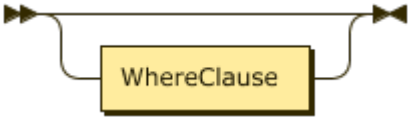

Figure 186: WhereClauseOptional

SelectStmtOpts:

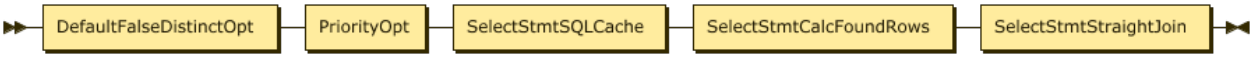

Figure 187: SelectStmtOpts

SelectStmtFieldList:

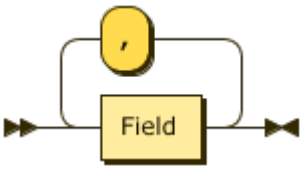

Figure 188: SelectStmtFieldList

TableRefsClause:

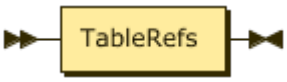

Figure 189: TableRefsClause

WhereClauseOptional:

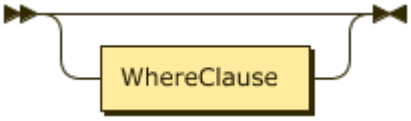

Figure 190: WhereClauseOptional

SelectStmtGroup:

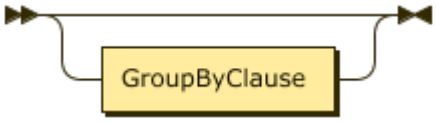

Figure 191: SelectStmtGroup

HavingClause:

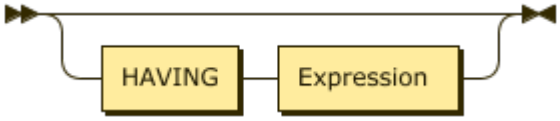

Figure 192: HavingClause

OrderByOptional:

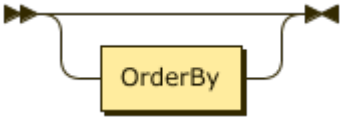

Figure 193: OrderByOptional

SelectStmtLimit:

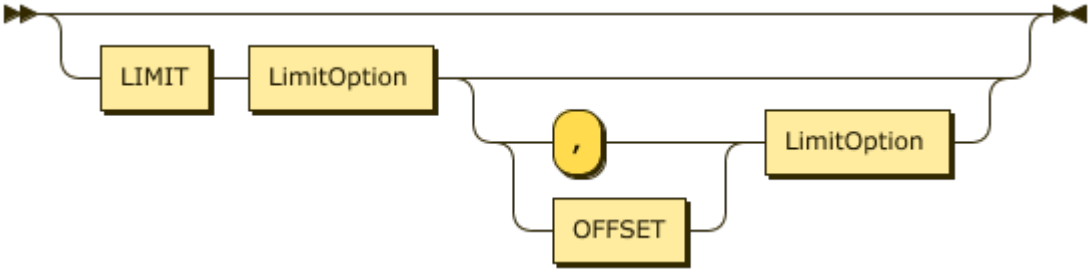

Figure 194: SelectStmtLimit

SelectLockOpt:

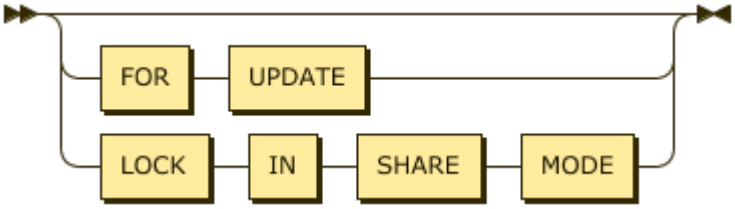

Figure 195: SelectLockOpt

4.1.5.55.2 Description of the syntax elements

Syntax Element	Description
`ALL`, `DISTINCT`, `DISTINCTROW`	The `ALL`, `DISTINCT`/`DISTINCTROW` modifiers specify whether duplicate rows should be returned. ALL (the default) specifies that all matching rows should be returned.
`HIGH_PRIORITY`	`HIGH_PRIORITY` gives the current statement higher priority than other statements.
`SQL_CALC_FOUND_ROWS`	To guarantee compatibility with MySQL, TiDB parses this syntax, but will ignore it.
`SQL_CACHE`, `SQL_NO_CACHE`	`SQL_CACHE` and `SQL_NO_CACHE` are used to control whether to cache the request results to the `BlockCache` of TiKV (RocksDB). For a one-time query on a large amount of data, such as the `count(*)` query, it is recommended to fill in `SQL_NO_CACHE` to avoid flushing the hot user data in `BlockCache`.
`STRAIGHT_JOIN`	`STRAIGHT_JOIN` forces the optimizer to do a union query in the order of the tables used in the `FROM` clause. When the optimizer chooses a join order that is not good, you can use this syntax to speed up the execution of the query.
`select_expr`	Each `select_expr` indicates a column to retrieve. including the column names and expressions. `\*` represents all the columns.
`FROM` ↪ `table_references`	The `FROM table_references` clause indicates the table (such as `select * from t;`), or tables (such as `select * from t1 join t2;`) or even 0 tables (such as `select 1+1 from dual;` which is equivalent to `select 1+1;`) from which to retrieve rows.
`WHERE` ↪ `where_condition`	The `WHERE` clause, if given, indicates the condition or conditions that rows must satisfy to be selected. The result contains only the data that meets the condition(s).
`GROUP BY`	The `GROUP BY` statement is used to group the result-set.
`HAVING` ↪ `where_condition`	The `HAVING` clause and the `WHERE` clause are both used to filter the results. The `HAVING` clause filters the results of `GROUP BY`, while the `WHERE` clause filter the results before aggregation.
`ORDER BY`	The `ORDER BY` clause is used to sort the data in ascending or descending order, based on columns, expressions or items in the `select_expr` list.

Syntax Element	Description
`LIMIT`	The `LIMIT` clause can be used to constrain the number of rows. `LIMIT` takes one or two numeric arguments. With one argument, the argument specifies the maximum number of rows to return, the first row to return is the first row of the table by default; with two arguments, the first argument specifies the offset of the first row to return, and the second specifies the maximum number of rows to return.
`FOR UPDATE`	The `SELECT FOR UPDATE` clause locks all the data in the result sets to detect concurrent updates from other transactions. Data that match the query conditions but do not exist in the result sets are not read-locked, such as the row data written by other transactions after the current transaction is started. TiDB uses the Optimistic Transaction Model. The transaction conflicts are not detected in the statement execution phase. Therefore, the current transaction does not block other transactions from executing `UPDATE`, `DELETE` or `SELECT FOR UPDATE` like other databases such as PostgreSQL. In the committing phase, the rows read by `SELECT FOR UPDATE` are committed in two phases, which means they can also join the conflict detection. If write conflicts occur, the commit fails for all transactions that include the `SELECT FOR UPDATE` clause. If no conflict is detected, the commit succeeds. And a new version is generated for the locked rows, so that write conflicts can be detected when other uncommitted transactions are being committed later. When using pessimistic transaction model, the behavior is basically the same as other databases. Refer to Difference with MySQL InnoDB to see the details.
`LOCK IN SHARE MODE`	To guarantee compatibility, TiDB parses these three modifiers, but will ignore them.

4.1.5.55.3 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 (c1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.03 sec)
Records: 5 Duplicates: 0 Warnings: 0
```

```
mysql> SELECT * FROM t1;
+----+----+
| id | c1 |
+----+----+
|  1 |  1 |
|  2 |  2 |
|  3 |  3 |
|  4 |  4 |
|  5 |  5 |
+----+----+
5 rows in set (0.00 sec)
```

4.1.5.55.4 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.55.5 See also

- INSERT
- DELETE
- UPDATE
- REPLACE

4.1.5.56 `SET DEFAULT ROLE`

This statement sets a specific role to be applied to a user by default. Thus, they will automatically have the permissions associated with a role without having to execute `SET ROLE <rolename>` or `SET ROLE ALL`.

4.1.5.56.1 Synopsis

SetDefaultRoleStmt:

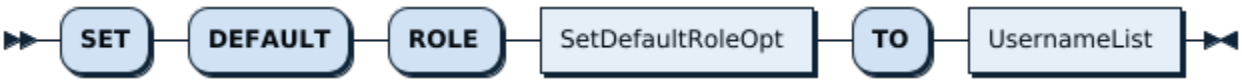

Figure 196: SetDefaultRoleStmt

SetDefaultRoleOpt:

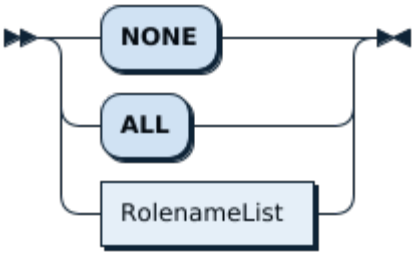

Figure 197: SetDefaultRoleOpt

RolenameList:

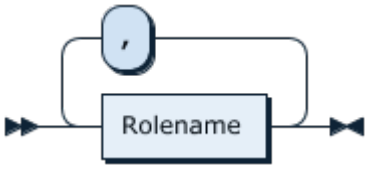

Figure 198: RolenameList

UsernameList:

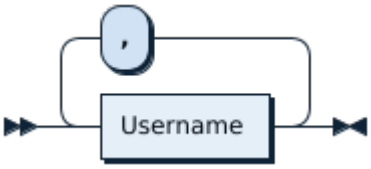

Figure 199: UsernameList

4.1.5.56.2 Examples

Create a new role for the analytics team, and a new user called `jennifer`:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 37
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> CREATE ROLE analyticsteam;
Query OK, 0 rows affected (0.02 sec)

mysql> GRANT SELECT ON test.* TO analyticsteam;
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE USER jennifer;
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT analyticsteam TO jennifer;
Query OK, 0 rows affected (0.01 sec)
```

Note that by default `jennifer` needs to `SET ROLE analyticsteam` in order to be able to use the privileges associated with the role:

```
$ mysql -ujennifer
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SHOW GRANTS;
+-----------------------------------------------+
| Grants for User                               |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'          |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+-----------------------------------------------+
2 rows in set (0.00 sec)

mysql> SHOW TABLES in test;
ERROR 1044 (42000): Access denied for user 'jennifer'@'%' to database 'test'
```

```
mysql> SET ROLE analyticsteam;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS;
+-------------------------------------------+
| Grants for User                           |
+-------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'      |
| GRANT Select ON test.* TO 'jennifer'@'%'  |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+-------------------------------------------+
3 rows in set (0.00 sec)

mysql> SHOW TABLES IN test;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)
```

The statement `SET DEFAULT ROLE` can be used to associated a role to `jennifer` so that she will not have to execute the statement `SET ROLE` in order to assume the privileges associated with the role:

```
$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 34
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SET DEFAULT ROLE analyticsteam TO jennifer;
Query OK, 0 rows affected (0.02 sec)
```

```
$ mysql -ujennifer
```

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 35
Server version: 5.7.25-TiDB-v4.0.0-beta.2-728-ga9177fe84 TiDB Server (Apache
    ↪  License 2.0) Community Edition, MySQL 5.7 compatible

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved
    ↪ .

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SHOW GRANTS;
+--------------------------------------------+
| Grants for User                            |
+--------------------------------------------+
| GRANT USAGE ON *.* TO 'jennifer'@'%'       |
| GRANT Select ON test.* TO 'jennifer'@'%'   |
| GRANT 'analyticsteam'@'%' TO 'jennifer'@'%' |
+--------------------------------------------+
3 rows in set (0.00 sec)

mysql> SHOW TABLES IN test;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)
```

`SET DEFAULT ROLE` will not automatically `GRANT` the associated role to the user. Attempting to `SET DEFAULT ROLE` for a role that `jennifer` does not have granted results in the following error:

```
mysql> SET DEFAULT ROLE analyticsteam TO jennifer;
ERROR 3530 (HY000): `analyticsteam`@`%` is is not granted to jennifer@%
```

4.1.5.56.3 MySQL compatibility

This statement is understood to be fully compatible with roles, which are a feature of MySQL 8.0. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.56.4 See also

- CREATE ROLE
- DROP ROLE
- `GRANT <role>`
- `REVOKE <role>`
- SET ROLE
- Role-Based Access Control

4.1.5.57 SET [NAMES|CHARACTER SET]

The statements `SET NAMES`, `SET CHARACTER SET` and `SET CHARSET` modify the variables `character_set_client`, `character_set_results` and `character_set_connection` for the current connection.

4.1.5.57.1 Synopsis

SetStmt:

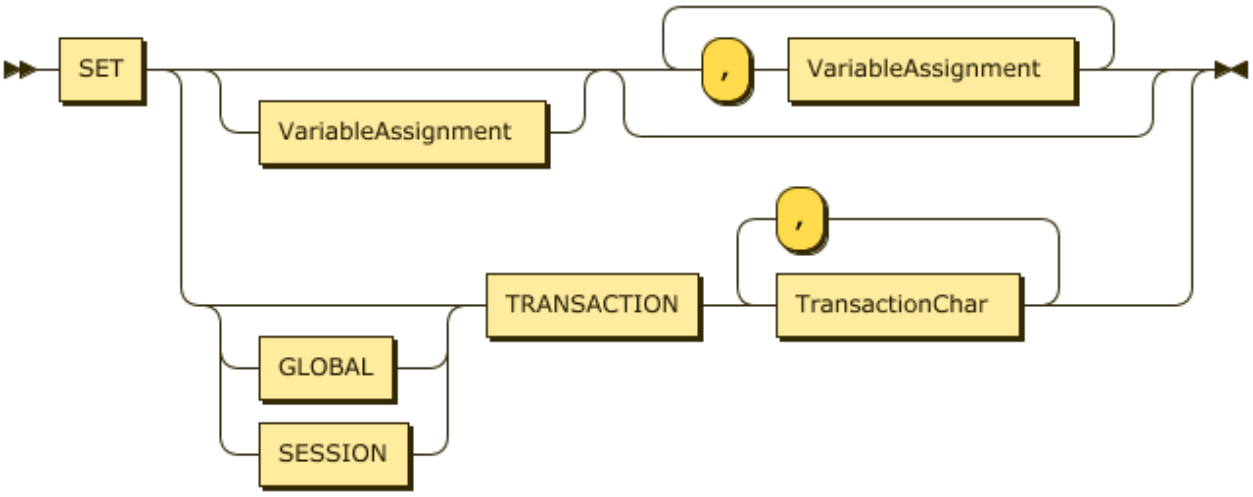

Figure 200: SetStmt

4.1.5.57.2 Examples

```
mysql> SHOW VARIABLES LIKE 'character_set%';
+--
    ↪ ----------------------+----------------------------------------------------------
    ↪
| Variable_name         | Value                                                    |
+--
    ↪ ----------------------+----------------------------------------------------------
    ↪
```

```
| character_sets_dir    | /usr/local/mysql-5.6.25-osx10.8-x86_64/share/
    ↪ charsets/ |
| character_set_connection | utf8mb4                                          |
| character_set_system  | utf8                                             |
| character_set_results | utf8mb4                                          |
| character_set_client  | utf8mb4                                          |
| character_set_database | utf8mb4                                         |
| character_set_filesystem | binary                                        |
| character_set_server  | utf8mb4                                          |
+--
    ↪ ----------------------+----------------------------------------------
    ↪
8 rows in set (0.01 sec)

mysql> SET NAMES utf8;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW VARIABLES LIKE 'character_set%';
+--
    ↪ ----------------------+----------------------------------------------
    ↪
| Variable_name         | Value                                          |
+--
    ↪ ----------------------+----------------------------------------------
    ↪
| character_sets_dir    | /usr/local/mysql-5.6.25-osx10.8-x86_64/share/
    ↪ charsets/ |
| character_set_connection | utf8                                          |
| character_set_system  | utf8                                             |
| character_set_results | utf8                                             |
| character_set_client  | utf8                                             |
| character_set_server  | utf8mb4                                          |
| character_set_database | utf8mb4                                         |
| character_set_filesystem | binary                                        |
+--
    ↪ ----------------------+----------------------------------------------
    ↪
8 rows in set (0.00 sec)

mysql> SET CHARACTER SET utf8mb4;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW VARIABLES LIKE 'character_set%';
+--
    ↪ ----------------------+----------------------------------------------
```

```
    ↪
| Variable_name          | Value                                                 |
+--
    ↪ ------------------------+-------------------------------------------------------
    ↪
| character_set_connection | utf8mb4                                                 |
| character_set_system   | utf8                                                    |
| character_set_results  | utf8mb4                                                 |
| character_set_client   | utf8mb4                                                 |
| character_sets_dir     | /usr/local/mysql-5.6.25-osx10.8-x86_64/share/
    ↪ charsets/ |
| character_set_database | utf8mb4                                                 |
| character_set_filesystem | binary                                                |
| character_set_server   | utf8mb4                                                 |
+--
    ↪ ------------------------+-------------------------------------------------------
    ↪
8 rows in set (0.00 sec)
```

4.1.5.57.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.57.4 See also

- SHOW [GLOBAL|SESSION] VARIABLES
- SET <variable>
- Character Set Support

4.1.5.58 SET PASSWORD

This statement changes the user password for a user account in the TiDB system database.

4.1.5.58.1 Synopsis

SetStmt:

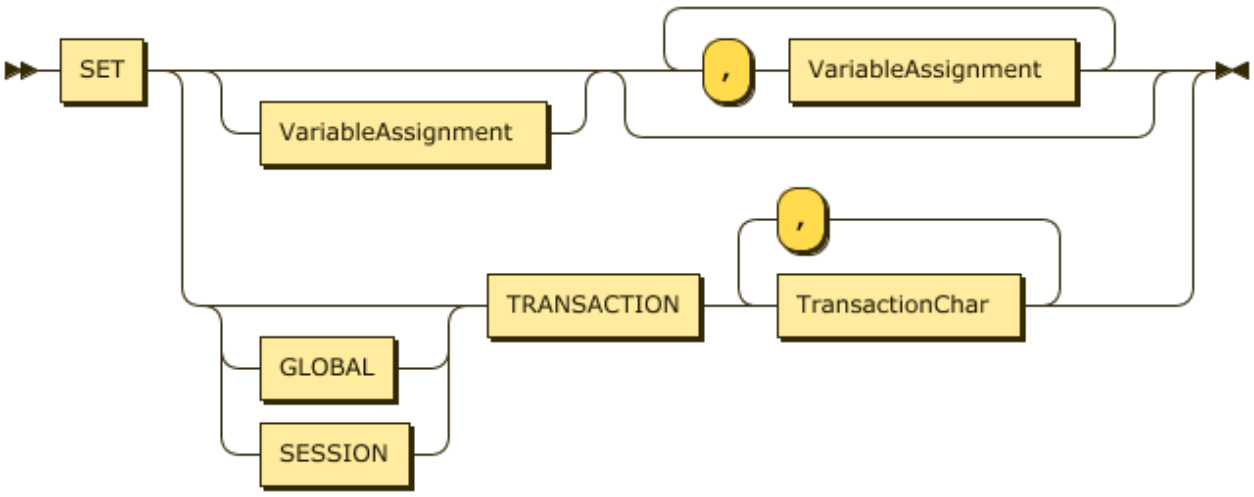

Figure 201: SetStmt

4.1.5.58.2 Examples

```
mysql> SET PASSWORD='test'; -- change my password
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE USER 'newuser' IDENTIFIED BY 'test';
Query OK, 1 row affected (0.00 sec)

mysql> SHOW CREATE USER newuser;
+--
    ↪ ------------------------------------------------------------------------------
    ↪
| CREATE USER for newuser@%
    ↪
    ↪ |
+--
    ↪ ------------------------------------------------------------------------------
    ↪
| CREATE USER 'newuser'@'%' IDENTIFIED WITH 'mysql_native_password' AS '*94
    ↪ BDCEBE19083CE2A1F959FD02F964C7AF4CFC29' REQUIRE NONE PASSWORD EXPIRE
    ↪ DEFAULT ACCOUNT UNLOCK |
+--
    ↪ ------------------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)

mysql> SET PASSWORD FOR newuser = 'test';
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> SHOW CREATE USER newuser;
+--
    ↪ ------------------------------------------------------------------------
    ↪
| CREATE USER for newuser@%
    ↪
    ↪ |
+--
    ↪ ------------------------------------------------------------------------
    ↪
| CREATE USER 'newuser'@'%' IDENTIFIED WITH 'mysql_native_password' AS '*94
    ↪ BDCEBE19083CE2A1F959FD02F964C7AF4CFC29' REQUIRE NONE PASSWORD EXPIRE
    ↪ DEFAULT ACCOUNT UNLOCK |
+--
    ↪ ------------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)

mysql> SET PASSWORD FOR newuser = PASSWORD('test'); -- deprecated syntax
    ↪ from earlier MySQL releases
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW CREATE USER newuser;
+--
    ↪ ------------------------------------------------------------------------
    ↪
| CREATE USER for newuser@%
    ↪
    ↪ |
+--
    ↪ ------------------------------------------------------------------------
    ↪
| CREATE USER 'newuser'@'%' IDENTIFIED WITH 'mysql_native_password' AS '*94
    ↪ BDCEBE19083CE2A1F959FD02F964C7AF4CFC29' REQUIRE NONE PASSWORD EXPIRE
    ↪ DEFAULT ACCOUNT UNLOCK |
+--
    ↪ ------------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)
```

4.1.5.58.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility

differences should be reported via an issue on GitHub.

4.1.5.58.4 See also

- CREATE USER
- Privilege Management

4.1.5.59 SET ROLE

The `SET ROLE` statement is used to enable roles in the current session. After enabling roles, users can use the privileges of the role(s).

4.1.5.59.1 Synopsis

SetRoleStmt:

Figure 202: SetRoleStmt

SetRoleOpt:

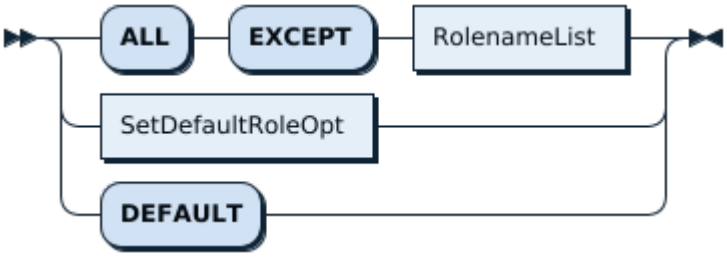

Figure 203: SetRoleOpt

SetDefaultRoleOpt:

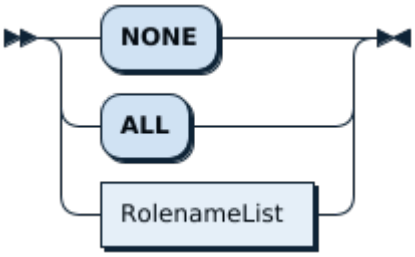

Figure 204: SetDefaultRoleOpt

4.1.5.59.2 Examples

Create a user `'u1'@'%'` and three roles: `'r1'@'%'`, `'r2'@'%'` and `'r3'@'%'`. Grant these roles to `'u1'@'%'` and set `'r1'@'%'` as the default role of `'u1'@'%'`.

```
CREATE USER 'u1'@'%';
CREATE ROLE 'r1', 'r2', 'r3';
GRANT 'r1', 'r2', 'r3' TO 'u1'@'%';
SET DEFAULT ROLE 'r1' TO 'u1'@'%';
```

Log in as `'u1'@'%'` and execute the following `SET ROLE` statement to enable all roles.

```
SET ROLE ALL;
SELECT CURRENT_ROLE();
```

```
+----------------------------+
| CURRENT_ROLE()             |
+----------------------------+
| `r1`@`%`,`r2`@`%`,`r3`@`%` |
+----------------------------+
1 row in set (0.000 sec)
```

Execute the following `SET ROLE` statement to enable `'r2'` and `'r3'`.

```
SET ROLE 'r2', 'r3';
SELECT CURRENT_ROLE();
```

```
+--------------------+
| CURRENT_ROLE()     |
+--------------------+
| `r2`@`%`,`r3`@`%` |
+--------------------+
1 row in set (0.000 sec)
```

Execute the following `SET ROLE` statement to enable the default role(s).

```
SET ROLE DEFAULT;
SELECT CURRENT_ROLE();
```

```
+----------------+
| CURRENT_ROLE() |
+----------------+
| `r1`@`%`       |
+----------------+
1 row in set (0.000 sec)
```

Execute the following `SET ROLE` statement to cancel all enabled role(s).

```
SET ROLE NONE;
SELECT CURRENT_ROLE();
```

```
+----------------+
| CURRENT_ROLE() |
+----------------+
|                |
+----------------+
1 row in set (0.000 sec)
```

4.1.5.59.3 MySQL compatibility

This statement is understood to be fully compatible with roles, which are a feature of MySQL 8.0. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.59.4 See also

- CREATE ROLE
- DROP ROLE
- `GRANT <role>`
- `REVOKE <role>`
- SET DEFAULT ROLE
- Role-Based Access Control

4.1.5.60 SET TRANSACTION

The `SET TRANSACTION` statement can be used to change the current isolation level on a `GLOBAL` or `SESSION` basis. This syntax is an alternative to `SET transaction_isolation='new-value'` and is included for compatibility with both MySQL, and the SQL standards.

4.1.5.60.1 Synopsis

SetStmt:

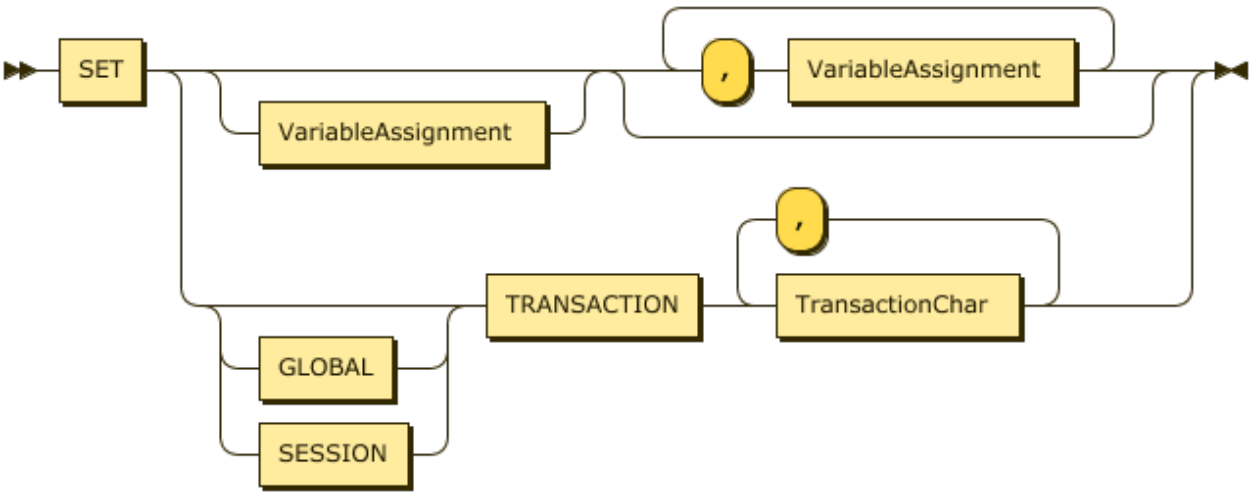

Figure 205: SetStmt

TransactionChar:

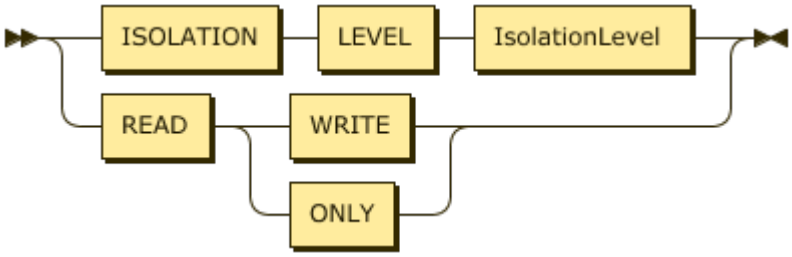

Figure 206: TransactionChar

IsolationLevel:

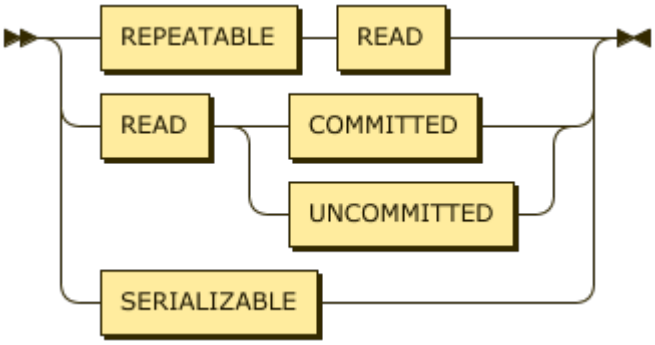

Figure 207: IsolationLevel

4.1.5.60.2 Examples

```
mysql> SHOW SESSION VARIABLES like 'transaction_isolation';
+-----------------------+----------------+
| Variable_name         | Value          |
+-----------------------+----------------+
| transaction_isolation | REPEATABLE-READ |
+-----------------------+----------------+
1 row in set (0.00 sec)

mysql> SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW SESSION VARIABLES like 'transaction_isolation';
+-----------------------+----------------+
| Variable_name         | Value          |
+-----------------------+----------------+
| transaction_isolation | READ-COMMITTED |
+-----------------------+----------------+
1 row in set (0.01 sec)

mysql> SET SESSION transaction_isolation = 'REPEATABLE-READ';
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW SESSION VARIABLES like 'transaction_isolation';
+-----------------------+----------------+
| Variable_name         | Value          |
+-----------------------+----------------+
| transaction_isolation | REPEATABLE-READ |
+-----------------------+----------------+
1 row in set (0.00 sec)
```

4.1.5.60.3 MySQL compatibility

- TiDB supports the ability to set a transaction as read-only in syntax only.
- The isolation levels `READ-UNCOMMITTED` and `SERIALIZABLE` are not supported.
- The isolation level `REPEATABLE-READ` is technically Snapshot Isolation. The name `REPEATABLE-READ` is used for compatibility with MySQL.

4.1.5.60.4 See also

- SET [GLOBAL|SESSION] <variable>
- Isolation Levels

4.1.5.61 SET [GLOBAL|SESSION] <variable>

The statement SET [GLOBAL|SESSION] modifies one of TiDB's built in variables, of either SESSION or GLOBAL scope. Note that similar to MySQL, changes to GLOBAL variables will not apply to either existing connections, or the local connection. Only new sessions will reflect the changes to the value.

4.1.5.61.1 Synopsis

SetStmt:

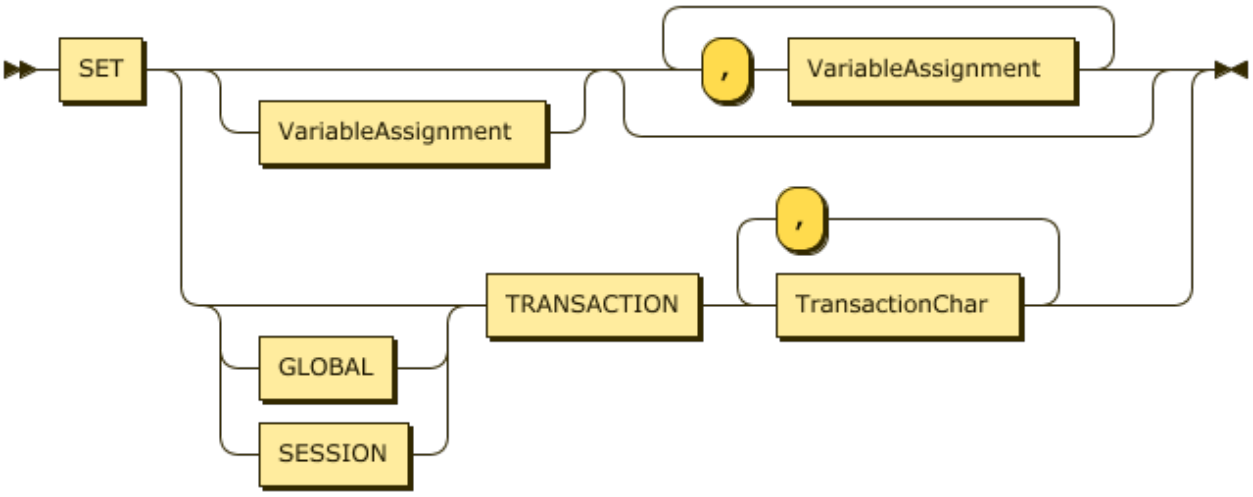

Figure 208: SetStmt

4.1.5.61.2 Examples

```
mysql> SHOW GLOBAL VARIABLES LIKE 'sql_mode';
+--
    ↪ -------------+-----------------------------------------------------------
    ↪
| Variable_name | Value
    ↪
    ↪ |
+--
    ↪ -------------+-----------------------------------------------------------
    ↪
| sql_mode      | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,
    ↪ NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,
    ↪ NO_ENGINE_SUBSTITUTION |
+--
    ↪ -------------+-----------------------------------------------------------
    ↪
```

```
1 row in set (0.00 sec)

mysql> SHOW SESSION VARIABLES LIKE 'sql_mode';
+--
    ↪ ------------+-----------------------------------------------------------
    ↪
| Variable_name | Value
    ↪
    ↪ |
+--
    ↪ ------------+-----------------------------------------------------------
    ↪
| sql_mode    | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,
    ↪ NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,
    ↪ NO_ENGINE_SUBSTITUTION |
+--
    ↪ ------------+-----------------------------------------------------------
    ↪
1 row in set (0.00 sec)

mysql> SET GLOBAL sql_mode = 'STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER';
Query OK, 0 rows affected (0.03 sec)

mysql> SHOW GLOBAL VARIABLES LIKE 'sql_mode';
+---------------+----------------------------------------+
| Variable_name | Value                                  |
+---------------+----------------------------------------+
| sql_mode    | STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER |
+---------------+----------------------------------------+
1 row in set (0.00 sec)

mysql> SHOW SESSION VARIABLES LIKE 'sql_mode';
+--
    ↪ ------------+-----------------------------------------------------------
    ↪
| Variable_name | Value
    ↪
    ↪ |
+--
    ↪ ------------+-----------------------------------------------------------
    ↪
| sql_mode    | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,
    ↪ NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,
    ↪ NO_ENGINE_SUBSTITUTION |
+--
```

```
    ↪ ------------+-------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)

mysql> SET SESSION sql_mode = 'STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER';
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW SESSION VARIABLES LIKE 'sql_mode';
+---------------+-----------------------------------------+
| Variable_name | Value                                   |
+---------------+-----------------------------------------+
| sql_mode      | STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER |
+---------------+-----------------------------------------+
1 row in set (0.00 sec)
```

4.1.5.61.3 MySQL compatibility

The following behavior differences apply:

- Changes made with `SET GLOBAL` will be propagated to all TiDB instances in the cluster. This differs from MySQL, where changes do not propagate to replicas.
- TiDB presents several variables as both readable and settable. This is required for MySQL compatibility, because it is common for both applications and connectors to read MySQL variables. For example: JDBC connectors both read and set query cache settings, despite not relying on the behavior.
- Changes made with `SET GLOBAL` will persist through TiDB server restarts. This means that `SET GLOBAL` in TiDB behaves more similar to `SET PERSIST` as available in MySQL 8.0 and above.

4.1.5.61.4 See also

- `SHOW [GLOBAL|SESSION] VARIABLES`

4.1.5.62 SHOW ANALYZE STATUS

The `SHOW ANALYZE STATUS` statement shows the statistics collection tasks being executed by TiDB and a limited number of historical task records.

4.1.5.62.1 Synopsis

ShowStmt:

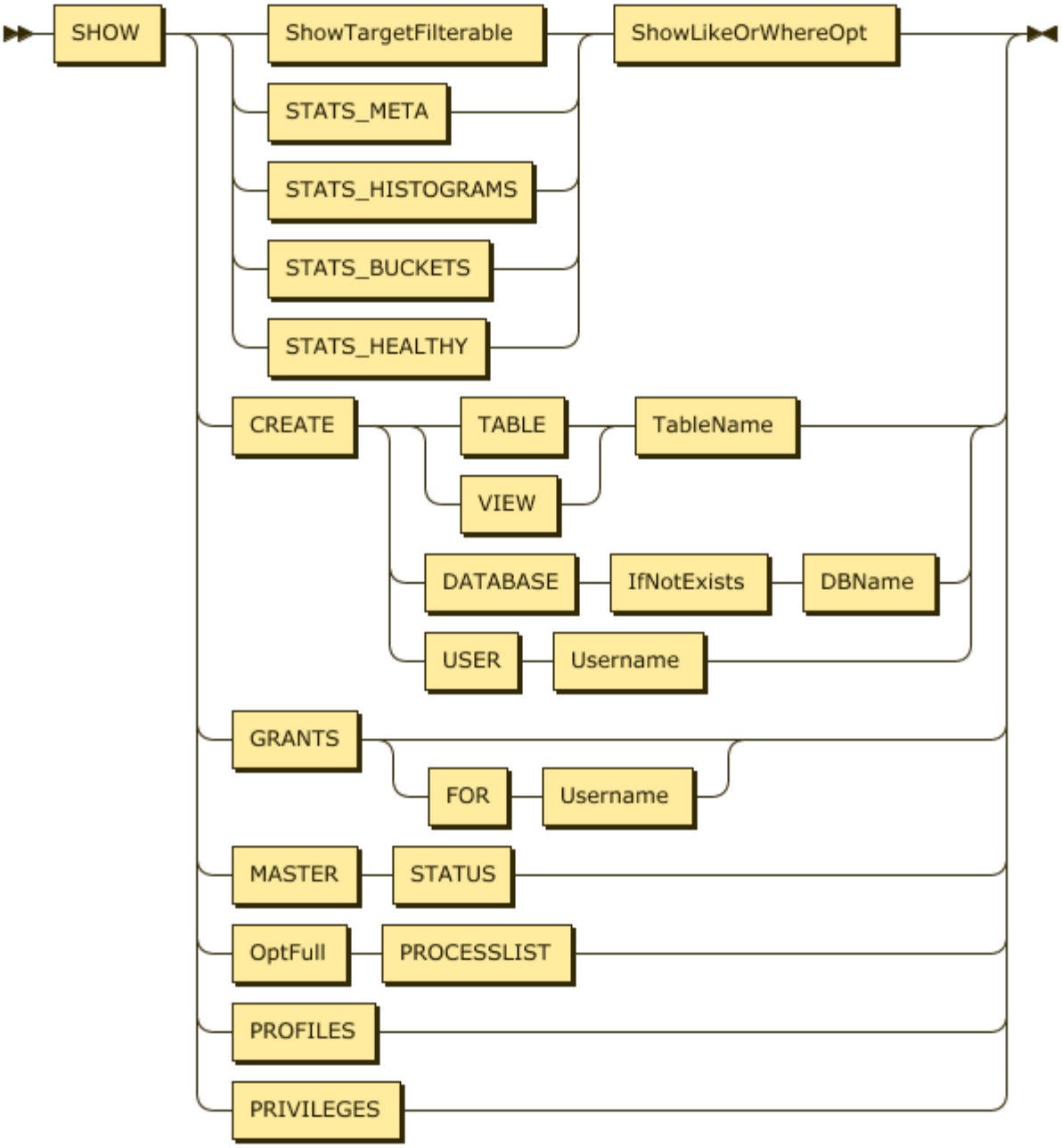

Figure 209: ShowStmt

ShowTargetFilterable:

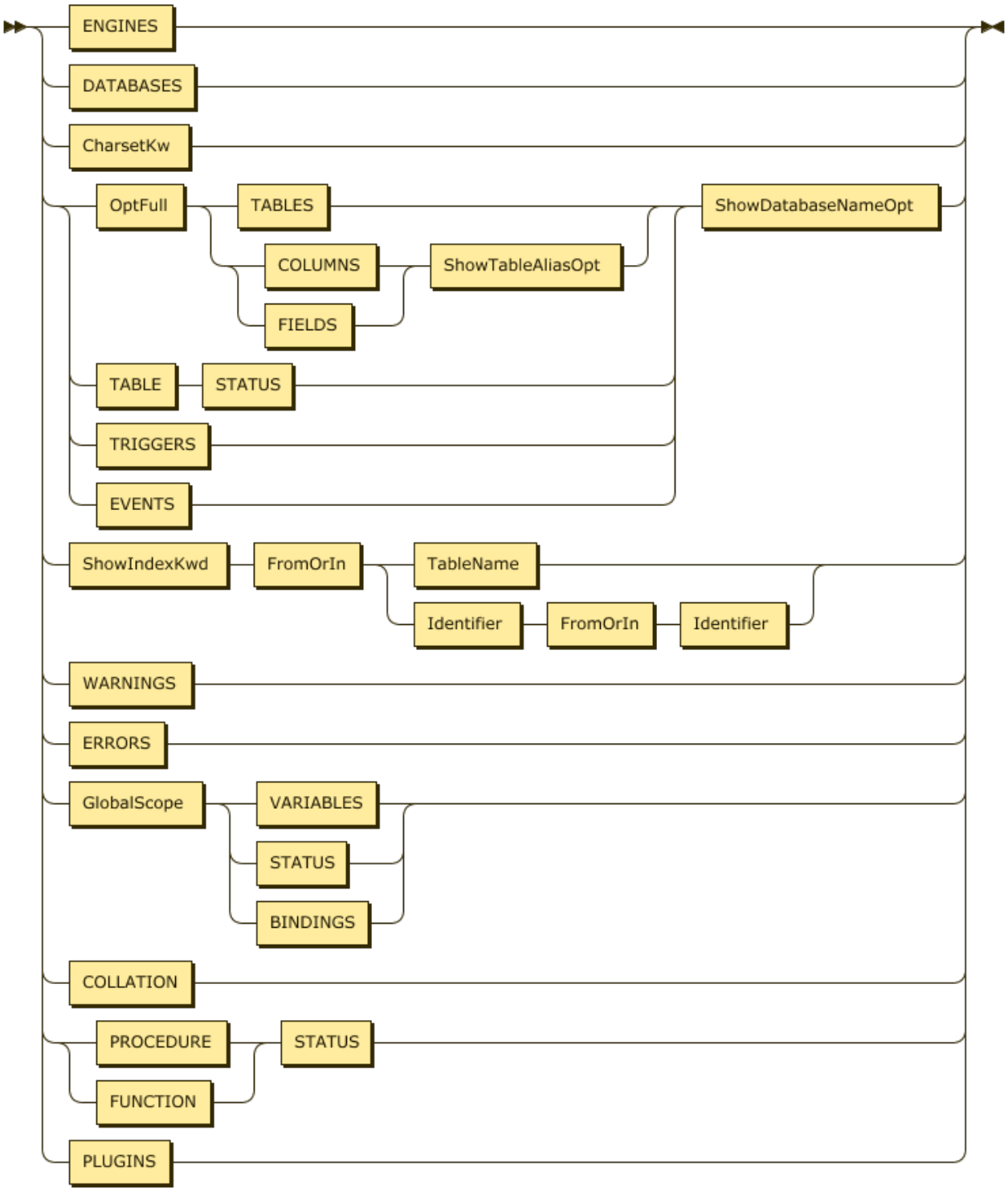

Figure 210: ShowTargetFilterable

4.1.5.62.2 Examples

```
create table t(x int, index idx(x)) partition by hash(x) partition 4;
```

```
analyze table t;
show analyze status;
```

```
+--
    ↪ -----------+-----------+---------------+------------------+----------------+
    ↪
| Table_schema | Table_name | Partition_name | Job_info  | Processed_rows |
    ↪ Start_time     | State   |
+--
    ↪ -----------+-----------+---------------+------------------+----------------+
    ↪
| test       | t         | p1           | analyze columns |            0 |
    ↪ 2020-05-25 17:23:55 | finished |
| test       | t         | p0           | analyze columns |            0 |
    ↪ 2020-05-25 17:23:55 | finished |
| test       | t         | p0           | analyze index idx |          0 |
    ↪ 2020-05-25 17:23:55 | finished |
| test       | t         | p1           | analyze index idx |          0 |
    ↪ 2020-05-25 17:23:55 | finished |
| test       | t         | p2           | analyze index idx |          0 |
    ↪ 2020-05-25 17:23:55 | finished |
| test       | t         | p3           | analyze index idx |          0 |
    ↪ 2020-05-25 17:23:55 | finished |
| test       | t         | p3           | analyze columns |            0 |
    ↪ 2020-05-25 17:23:55 | finished |
| test       | t         | p2           | analyze columns |            0 |
    ↪ 2020-05-25 17:23:55 | finished |
+--
    ↪ -----------+-----------+---------------+------------------+----------------+
    ↪
8 rows in set (0.00 sec)
```

4.1.5.62.3 MySQL compatibility

This statement is a TiDB extension to MySQL syntax.

4.1.5.62.4 See also

- ANALYZE_STATUS table

4.1.5.63 SHOW CHARACTER SET

This statement provides a static list of available character sets in TiDB. The output does not reflect any attributes of the current connection or user.

4.1.5.63.1 Synopsis

ShowStmt:

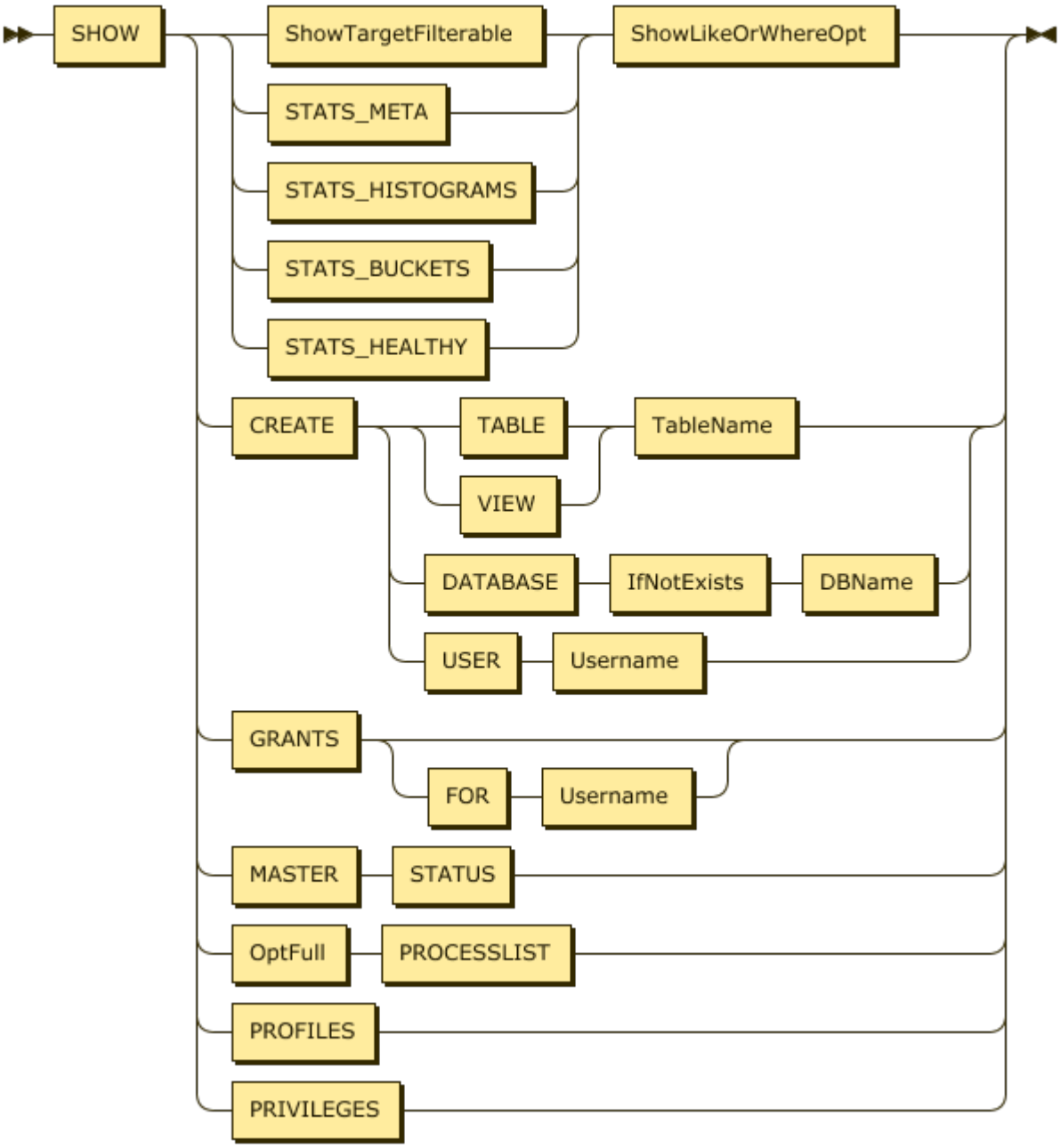

Figure 211: ShowStmt

ShowTargetFilterable:

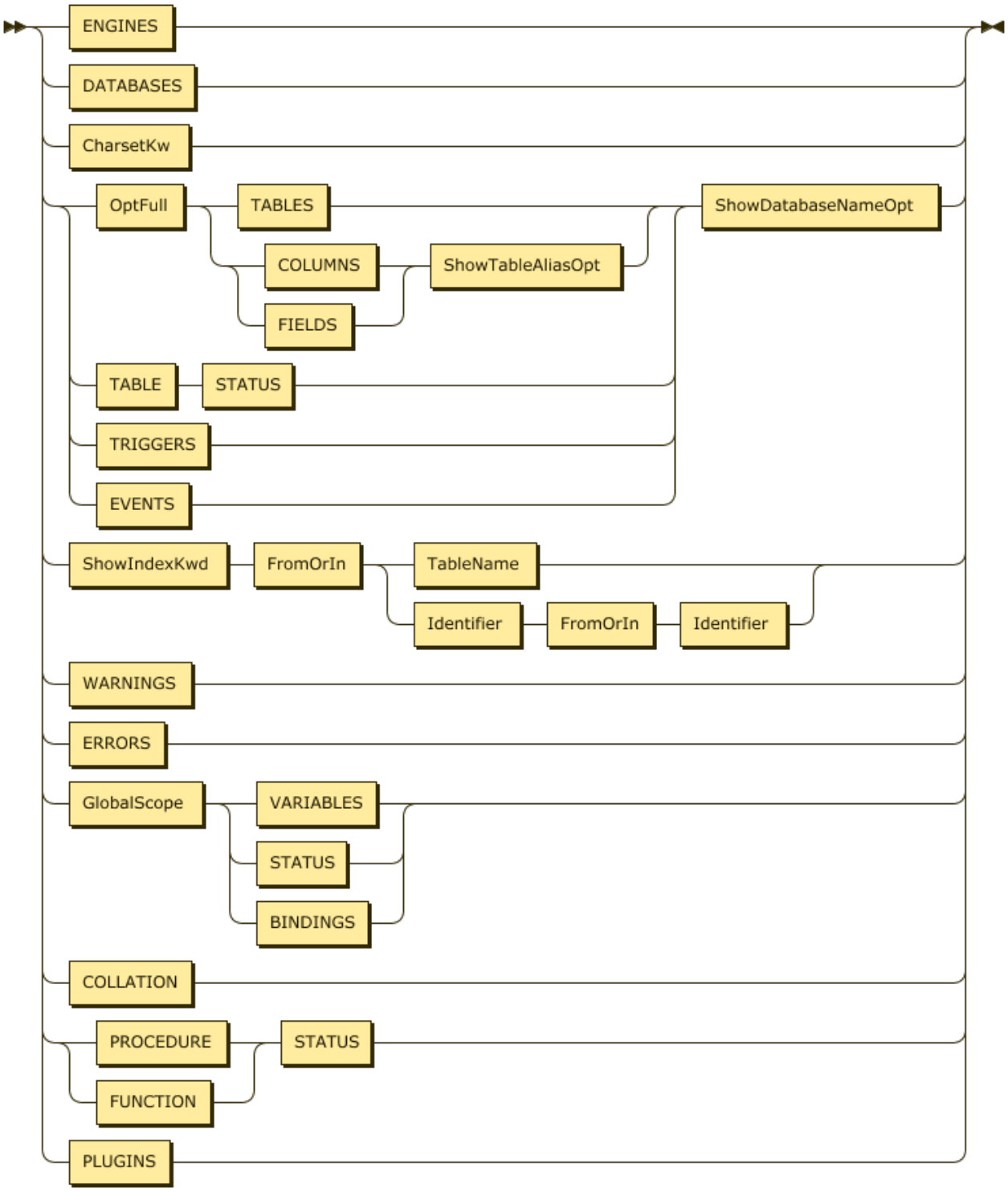

Figure 212: ShowTargetFilterable

CharsetKw:

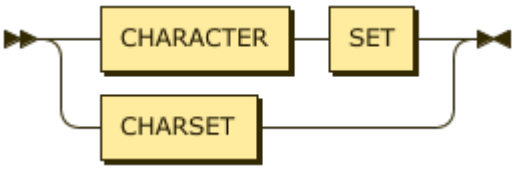

Figure 213: CharsetKw

4.1.5.63.2 Examples

```
mysql> SHOW CHARACTER SET;
+---------+---------------+--------------------+--------+
| Charset | Description | Default collation | Maxlen |
+---------+---------------+--------------------+--------+
| utf8   | UTF-8 Unicode | utf8_bin      |     3 |
| utf8mb4 | UTF-8 Unicode | utf8mb4_bin |     4 |
| ascii  | US ASCII     | ascii_bin       |     1 |
| latin1 | Latin1       | latin1_bin      |     1 |
| binary | binary       | binary          |     1 |
+---------+---------------+--------------------+--------+
5 rows in set (0.00 sec)
```

4.1.5.63.3 MySQL compatibility

The usage of this statement is understood to be fully compatible with MySQL. However, charsets in TiDB may have different default collations compared with MySQL. For details, refer to Compatibility with MySQL. Any other compatibility differences should be reported via an issue on GitHub.

4.1.5.63.4 See also

- SHOW COLLATION

4.1.5.64 SHOW COLLATION

This statement provides a static list of collations, and is included to provide compatibility with MySQL client libraries.

> **Note:** TiDB currently only supports binary collations.

4.1.5.64.1 Synopsis

ShowStmt:

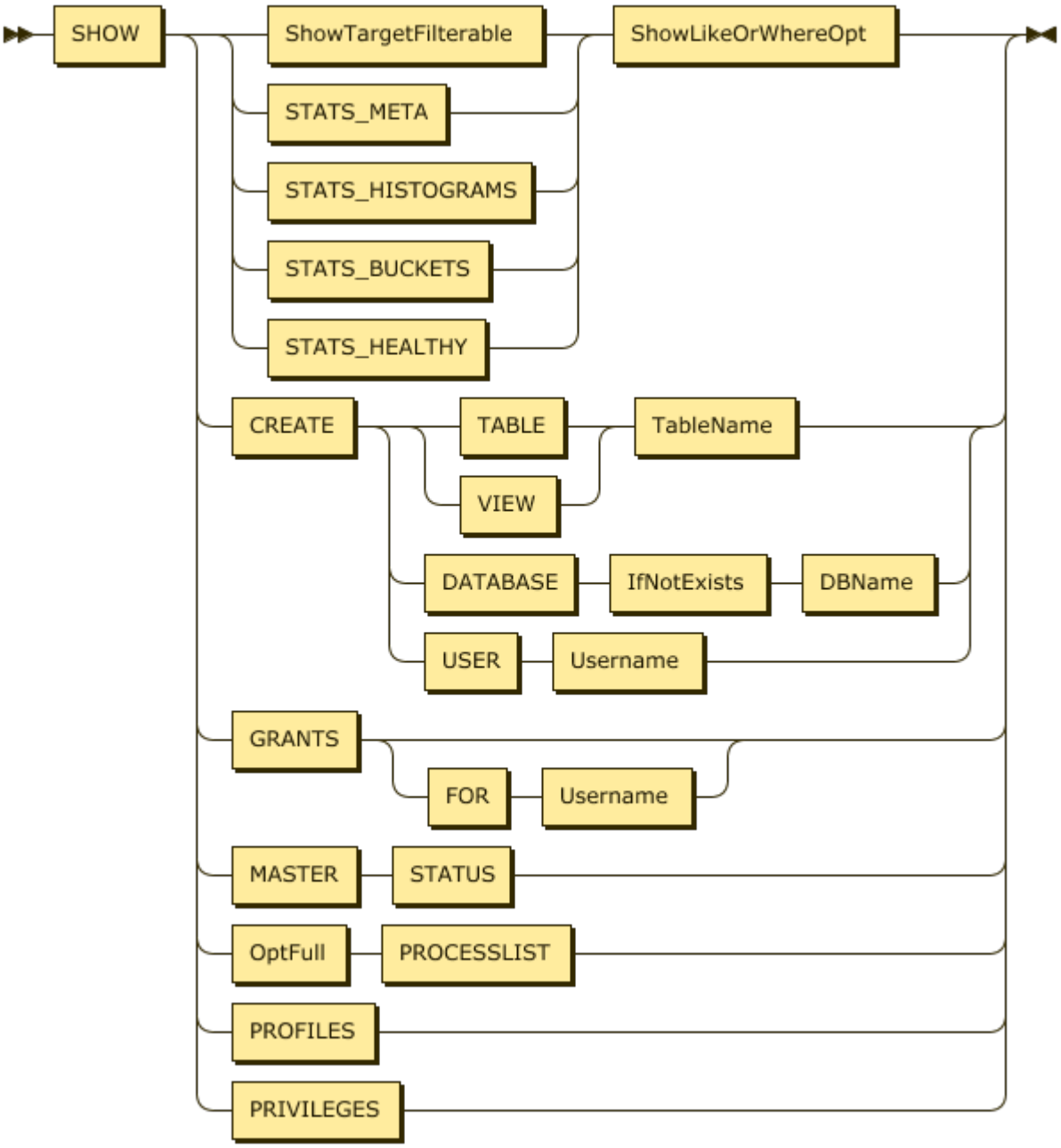

Figure 214: ShowStmt

ShowTargetFilterable:

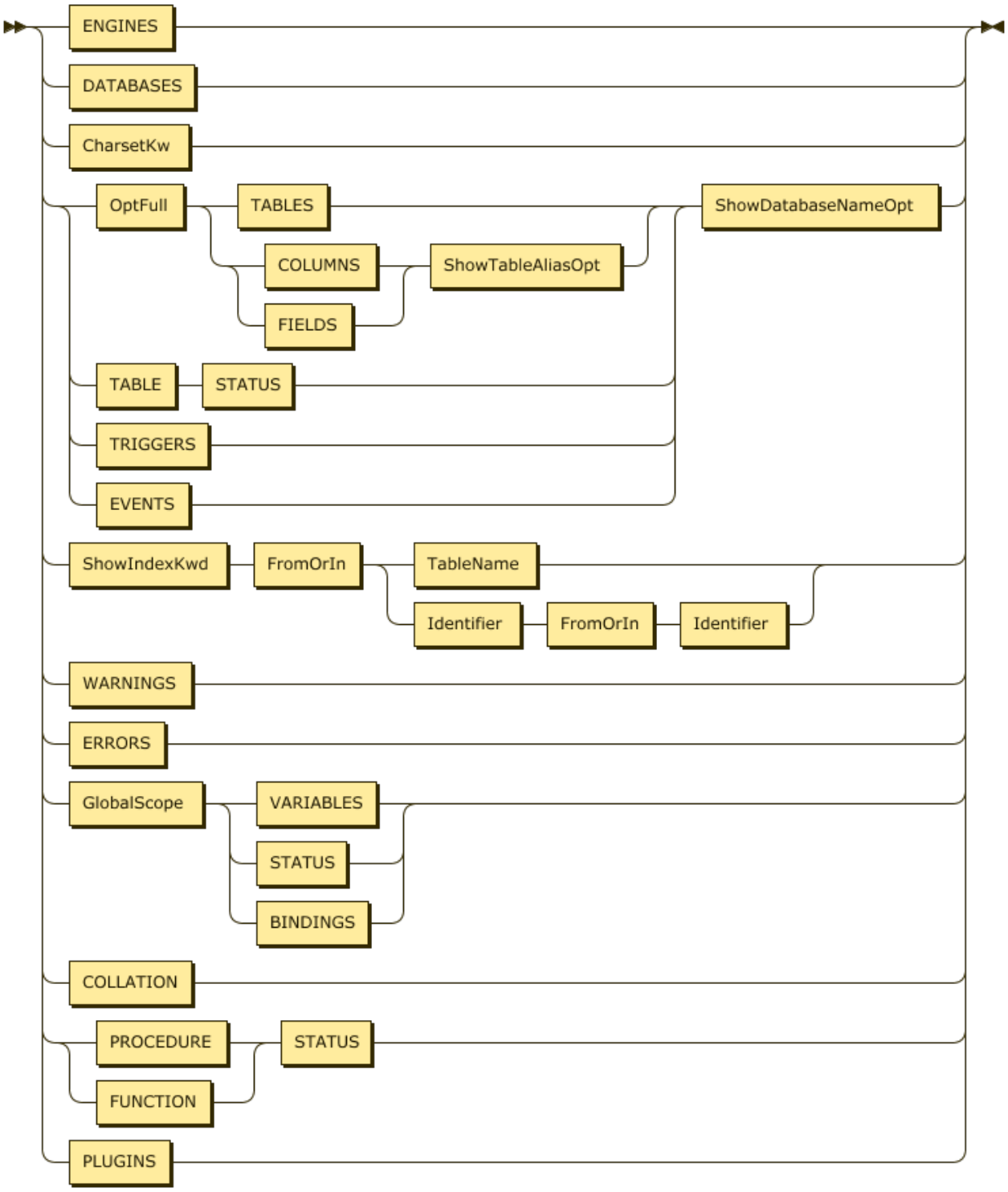

Figure 215: ShowTargetFilterable

4.1.5.64.2 Examples

```
mysql> SHOW COLLATION;
```

```
+--
    ↪ ----------------------+----------+------+---------+----------+----------+
    ↪
| Collation              | Charset | Id  | Default | Compiled | Sortlen |
+--
    ↪ ----------------------+----------+------+---------+----------+----------+
    ↪
| big5_chinese_ci        | big5     |   1 | Yes     | Yes      |       1 |
| latin2_czech_cs        | latin2   |   2 |         | Yes      |       1 |
| dec8_swedish_ci        | dec8     |   3 | Yes     | Yes      |       1 |
| cp850_general_ci       | cp850    |   4 | Yes     | Yes      |       1 |
| latin1_german1_ci      | latin1   |   5 |         | Yes      |       1 |
| hp8_english_ci         | hp8      |   6 | Yes     | Yes      |       1 |
| koi8r_general_ci       | koi8r    |   7 | Yes     | Yes      |       1 |
| latin1_swedish_ci      | latin1   |   8 | Yes     | Yes      |       1 |
| latin2_general_ci      | latin2   |   9 | Yes     | Yes      |       1 |
| swe7_swedish_ci        | swe7     |  10 | Yes     | Yes      |       1 |
| ascii_general_ci       | ascii    |  11 | Yes     | Yes      |       1 |
| ujis_japanese_ci       | ujis     |  12 | Yes     | Yes      |       1 |
| sjis_japanese_ci       | sjis     |  13 | Yes     | Yes      |       1 |
| cp1251_bulgarian_ci    | cp1251   |  14 |         | Yes      |       1 |
| latin1_danish_ci       | latin1   |  15 |         | Yes      |       1 |
| hebrew_general_ci      | hebrew   |  16 | Yes     | Yes      |       1 |
| tis620_thai_ci         | tis620   |  18 | Yes     | Yes      |       1 |
| euckr_korean_ci        | euckr    |  19 | Yes     | Yes      |       1 |
| latin7_estonian_cs     | latin7   |  20 |         | Yes      |       1 |
| latin2_hungarian_ci    | latin2   |  21 |         | Yes      |       1 |
| koi8u_general_ci       | koi8u    |  22 | Yes     | Yes      |       1 |
| cp1251_ukrainian_ci    | cp1251   |  23 |         | Yes      |       1 |
| gb2312_chinese_ci      | gb2312   |  24 | Yes     | Yes      |       1 |
| greek_general_ci       | greek    |  25 | Yes     | Yes      |       1 |
| cp1250_general_ci      | cp1250   |  26 | Yes     | Yes      |       1 |
| latin2_croatian_ci     | latin2   |  27 |         | Yes      |       1 |
| gbk_chinese_ci         | gbk      |  28 | Yes     | Yes      |       1 |
| cp1257_lithuanian_ci   | cp1257   |  29 |         | Yes      |       1 |
| latin5_turkish_ci      | latin5   |  30 | Yes     | Yes      |       1 |
| latin1_german2_ci      | latin1   |  31 |         | Yes      |       1 |
| armscii8_general_ci    | armscii8 |  32 | Yes     | Yes      |       1 |
| utf8_general_ci        | utf8     |  33 | Yes     | Yes      |       1 |
| cp1250_czech_cs        | cp1250   |  34 |         | Yes      |       1 |
| ucs2_general_ci        | ucs2     |  35 | Yes     | Yes      |       1 |
| cp866_general_ci       | cp866    |  36 | Yes     | Yes      |       1 |
| keybcs2_general_ci     | keybcs2  |  37 | Yes     | Yes      |       1 |
| macce_general_ci       | macce    |  38 | Yes     | Yes      |       1 |
| macroman_general_ci    | macroman |  39 | Yes     | Yes      |       1 |
```

```
| cp852_general_ci     | cp852   |  40 | Yes |  Yes |       1 |
| latin7_general_ci    | latin7  |  41 | Yes |  Yes |       1 |
| latin7_general_cs    | latin7  |  42 |     |  Yes |       1 |
| macce_bin            | macce   |  43 |     |  Yes |       1 |
| cp1250_croatian_ci   | cp1250  |  44 |     |  Yes |       1 |
| utf8mb4_general_ci   | utf8mb4 |  45 | Yes |  Yes |       1 |
| utf8mb4_bin          | utf8mb4 |  46 |     |  Yes |       1 |
| latin1_bin           | latin1  |  47 |     |  Yes |       1 |
| latin1_general_ci    | latin1  |  48 |     |  Yes |       1 |
| latin1_general_cs    | latin1  |  49 |     |  Yes |       1 |
| cp1251_bin           | cp1251  |  50 |     |  Yes |       1 |
| cp1251_general_ci    | cp1251  |  51 | Yes |  Yes |       1 |
| cp1251_general_cs    | cp1251  |  52 |     |  Yes |       1 |
| macroman_bin         | macroman |  53 |     |  Yes |       1 |
| utf16_general_ci     | utf16   |  54 | Yes |  Yes |       1 |
| utf16_bin            | utf16   |  55 |     |  Yes |       1 |
| utf16le_general_ci   | utf16le |  56 | Yes |  Yes |       1 |
| cp1256_general_ci    | cp1256  |  57 | Yes |  Yes |       1 |
| cp1257_bin           | cp1257  |  58 |     |  Yes |       1 |
| cp1257_general_ci    | cp1257  |  59 | Yes |  Yes |       1 |
| utf32_general_ci     | utf32   |  60 | Yes |  Yes |       1 |
| utf32_bin            | utf32   |  61 |     |  Yes |       1 |
| utf16le_bin          | utf16le |  62 |     |  Yes |       1 |
| binary               | binary  |  63 | Yes |  Yes |       1 |
| armscii8_bin         | armscii8 |  64 |     |  Yes |       1 |
| ascii_bin            | ascii   |  65 |     |  Yes |       1 |
| cp1250_bin           | cp1250  |  66 |     |  Yes |       1 |
| cp1256_bin           | cp1256  |  67 |     |  Yes |       1 |
| cp866_bin            | cp866   |  68 |     |  Yes |       1 |
| dec8_bin             | dec8    |  69 |     |  Yes |       1 |
| greek_bin            | greek   |  70 |     |  Yes |       1 |
| hebrew_bin           | hebrew  |  71 |     |  Yes |       1 |
| hp8_bin              | hp8     |  72 |     |  Yes |       1 |
| keybcs2_bin          | keybcs2 |  73 |     |  Yes |       1 |
| koi8r_bin            | koi8r   |  74 |     |  Yes |       1 |
| koi8u_bin            | koi8u   |  75 |     |  Yes |       1 |
| latin2_bin           | latin2  |  77 |     |  Yes |       1 |
| latin5_bin           | latin5  |  78 |     |  Yes |       1 |
| latin7_bin           | latin7  |  79 |     |  Yes |       1 |
| cp850_bin            | cp850   |  80 |     |  Yes |       1 |
| cp852_bin            | cp852   |  81 |     |  Yes |       1 |
| swe7_bin             | swe7    |  82 |     |  Yes |       1 |
| utf8_bin             | utf8    |  83 |     |  Yes |       1 |
| big5_bin             | big5    |  84 |     |  Yes |       1 |
| euckr_bin            | euckr   |  85 |     |  Yes |       1 |
```

gb2312_bin	gb2312	86		Yes	1
gbk_bin	gbk	87		Yes	1
sjis_bin	sjis	88		Yes	1
tis620_bin	tis620	89		Yes	1
ucs2_bin	ucs2	90		Yes	1
ujis_bin	ujis	91		Yes	1
geostd8_general_ci	geostd8	92	Yes	Yes	1
geostd8_bin	geostd8	93		Yes	1
latin1_spanish_ci	latin1	94		Yes	1
cp932_japanese_ci	cp932	95	Yes	Yes	1
cp932_bin	cp932	96		Yes	1
eucjpms_japanese_ci	eucjpms	97	Yes	Yes	1
eucjpms_bin	eucjpms	98		Yes	1
cp1250_polish_ci	cp1250	99		Yes	1
utf16_unicode_ci	utf16	101		Yes	1
utf16_icelandic_ci	utf16	102		Yes	1
utf16_latvian_ci	utf16	103		Yes	1
utf16_romanian_ci	utf16	104		Yes	1
utf16_slovenian_ci	utf16	105		Yes	1
utf16_polish_ci	utf16	106		Yes	1
utf16_estonian_ci	utf16	107		Yes	1
utf16_spanish_ci	utf16	108		Yes	1
utf16_swedish_ci	utf16	109		Yes	1
utf16_turkish_ci	utf16	110		Yes	1
utf16_czech_ci	utf16	111		Yes	1
utf16_danish_ci	utf16	112		Yes	1
utf16_lithuanian_ci	utf16	113		Yes	1
utf16_slovak_ci	utf16	114		Yes	1
utf16_spanish2_ci	utf16	115		Yes	1
utf16_roman_ci	utf16	116		Yes	1
utf16_persian_ci	utf16	117		Yes	1
utf16_esperanto_ci	utf16	118		Yes	1
utf16_hungarian_ci	utf16	119		Yes	1
utf16_sinhala_ci	utf16	120		Yes	1
utf16_german2_ci	utf16	121		Yes	1
utf16_croatian_ci	utf16	122		Yes	1
utf16_unicode_520_ci	utf16	123		Yes	1
utf16_vietnamese_ci	utf16	124		Yes	1
ucs2_unicode_ci	ucs2	128		Yes	1
ucs2_icelandic_ci	ucs2	129		Yes	1
ucs2_latvian_ci	ucs2	130		Yes	1
ucs2_romanian_ci	ucs2	131		Yes	1
ucs2_slovenian_ci	ucs2	132		Yes	1
ucs2_polish_ci	ucs2	133		Yes	1
ucs2_estonian_ci	ucs2	134		Yes	1

```
| ucs2_spanish_ci       | ucs2    |  135 |         | Yes     |       1 |
| ucs2_swedish_ci       | ucs2    |  136 |         | Yes     |       1 |
| ucs2_turkish_ci       | ucs2    |  137 |         | Yes     |       1 |
| ucs2_czech_ci         | ucs2    |  138 |         | Yes     |       1 |
| ucs2_danish_ci        | ucs2    |  139 |         | Yes     |       1 |
| ucs2_lithuanian_ci    | ucs2    |  140 |         | Yes     |       1 |
| ucs2_slovak_ci        | ucs2    |  141 |         | Yes     |       1 |
| ucs2_spanish2_ci      | ucs2    |  142 |         | Yes     |       1 |
| ucs2_roman_ci         | ucs2    |  143 |         | Yes     |       1 |
| ucs2_persian_ci       | ucs2    |  144 |         | Yes     |       1 |
| ucs2_esperanto_ci     | ucs2    |  145 |         | Yes     |       1 |
| ucs2_hungarian_ci     | ucs2    |  146 |         | Yes     |       1 |
| ucs2_sinhala_ci       | ucs2    |  147 |         | Yes     |       1 |
| ucs2_german2_ci       | ucs2    |  148 |         | Yes     |       1 |
| ucs2_croatian_ci      | ucs2    |  149 |         | Yes     |       1 |
| ucs2_unicode_520_ci   | ucs2    |  150 |         | Yes     |       1 |
| ucs2_vietnamese_ci    | ucs2    |  151 |         | Yes     |       1 |
| ucs2_general_mysql500_ci | ucs2 | 159 |         | Yes     |       1 |
| utf32_unicode_ci      | utf32   |  160 |         | Yes     |       1 |
| utf32_icelandic_ci    | utf32   |  161 |         | Yes     |       1 |
| utf32_latvian_ci      | utf32   |  162 |         | Yes     |       1 |
| utf32_romanian_ci     | utf32   |  163 |         | Yes     |       1 |
| utf32_slovenian_ci    | utf32   |  164 |         | Yes     |       1 |
| utf32_polish_ci       | utf32   |  165 |         | Yes     |       1 |
| utf32_estonian_ci     | utf32   |  166 |         | Yes     |       1 |
| utf32_spanish_ci      | utf32   |  167 |         | Yes     |       1 |
| utf32_swedish_ci      | utf32   |  168 |         | Yes     |       1 |
| utf32_turkish_ci      | utf32   |  169 |         | Yes     |       1 |
| utf32_czech_ci        | utf32   |  170 |         | Yes     |       1 |
| utf32_danish_ci       | utf32   |  171 |         | Yes     |       1 |
| utf32_lithuanian_ci   | utf32   |  172 |         | Yes     |       1 |
| utf32_slovak_ci       | utf32   |  173 |         | Yes     |       1 |
| utf32_spanish2_ci     | utf32   |  174 |         | Yes     |       1 |
| utf32_roman_ci        | utf32   |  175 |         | Yes     |       1 |
| utf32_persian_ci      | utf32   |  176 |         | Yes     |       1 |
| utf32_esperanto_ci    | utf32   |  177 |         | Yes     |       1 |
| utf32_hungarian_ci    | utf32   |  178 |         | Yes     |       1 |
| utf32_sinhala_ci      | utf32   |  179 |         | Yes     |       1 |
| utf32_german2_ci      | utf32   |  180 |         | Yes     |       1 |
| utf32_croatian_ci     | utf32   |  181 |         | Yes     |       1 |
| utf32_unicode_520_ci  | utf32   |  182 |         | Yes     |       1 |
| utf32_vietnamese_ci   | utf32   |  183 |         | Yes     |       1 |
| utf8_unicode_ci       | utf8    |  192 |         | Yes     |       1 |
| utf8_icelandic_ci     | utf8    |  193 |         | Yes     |       1 |
| utf8_latvian_ci       | utf8    |  194 |         | Yes     |       1 |
```

```
| utf8_romanian_ci      | utf8    |  195 |         | Yes     |       1 |
| utf8_slovenian_ci     | utf8    |  196 |         | Yes     |       1 |
| utf8_polish_ci        | utf8    |  197 |         | Yes     |       1 |
| utf8_estonian_ci      | utf8    |  198 |         | Yes     |       1 |
| utf8_spanish_ci       | utf8    |  199 |         | Yes     |       1 |
| utf8_swedish_ci       | utf8    |  200 |         | Yes     |       1 |
| utf8_turkish_ci       | utf8    |  201 |         | Yes     |       1 |
| utf8_czech_ci         | utf8    |  202 |         | Yes     |       1 |
| utf8_danish_ci        | utf8    |  203 |         | Yes     |       1 |
| utf8_lithuanian_ci    | utf8    |  204 |         | Yes     |       1 |
| utf8_slovak_ci        | utf8    |  205 |         | Yes     |       1 |
| utf8_spanish2_ci      | utf8    |  206 |         | Yes     |       1 |
| utf8_roman_ci         | utf8    |  207 |         | Yes     |       1 |
| utf8_persian_ci       | utf8    |  208 |         | Yes     |       1 |
| utf8_esperanto_ci     | utf8    |  209 |         | Yes     |       1 |
| utf8_hungarian_ci     | utf8    |  210 |         | Yes     |       1 |
| utf8_sinhala_ci       | utf8    |  211 |         | Yes     |       1 |
| utf8_german2_ci       | utf8    |  212 |         | Yes     |       1 |
| utf8_croatian_ci      | utf8    |  213 |         | Yes     |       1 |
| utf8_unicode_520_ci   | utf8    |  214 |         | Yes     |       1 |
| utf8_vietnamese_ci    | utf8    |  215 |         | Yes     |       1 |
| utf8_general_mysql500_ci | utf8 | 223 |         | Yes     |       1 |
| utf8mb4_unicode_ci    | utf8mb4 |  224 |         | Yes     |       1 |
| utf8mb4_icelandic_ci  | utf8mb4 |  225 |         | Yes     |       1 |
| utf8mb4_latvian_ci    | utf8mb4 |  226 |         | Yes     |       1 |
| utf8mb4_romanian_ci   | utf8mb4 |  227 |         | Yes     |       1 |
| utf8mb4_slovenian_ci  | utf8mb4 |  228 |         | Yes     |       1 |
| utf8mb4_polish_ci     | utf8mb4 |  229 |         | Yes     |       1 |
| utf8mb4_estonian_ci   | utf8mb4 |  230 |         | Yes     |       1 |
| utf8mb4_spanish_ci    | utf8mb4 |  231 |         | Yes     |       1 |
| utf8mb4_swedish_ci    | utf8mb4 |  232 |         | Yes     |       1 |
| utf8mb4_turkish_ci    | utf8mb4 |  233 |         | Yes     |       1 |
| utf8mb4_czech_ci      | utf8mb4 |  234 |         | Yes     |       1 |
| utf8mb4_danish_ci     | utf8mb4 |  235 |         | Yes     |       1 |
| utf8mb4_lithuanian_ci | utf8mb4 |  236 |         | Yes     |       1 |
| utf8mb4_slovak_ci     | utf8mb4 |  237 |         | Yes     |       1 |
| utf8mb4_spanish2_ci   | utf8mb4 |  238 |         | Yes     |       1 |
| utf8mb4_roman_ci      | utf8mb4 |  239 |         | Yes     |       1 |
| utf8mb4_persian_ci    | utf8mb4 |  240 |         | Yes     |       1 |
| utf8mb4_esperanto_ci  | utf8mb4 |  241 |         | Yes     |       1 |
| utf8mb4_hungarian_ci  | utf8mb4 |  242 |         | Yes     |       1 |
| utf8mb4_sinhala_ci    | utf8mb4 |  243 |         | Yes     |       1 |
| utf8mb4_german2_ci    | utf8mb4 |  244 |         | Yes     |       1 |
| utf8mb4_croatian_ci   | utf8mb4 |  245 |         | Yes     |       1 |
| utf8mb4_unicode_520_ci | utf8mb4 | 246 |         | Yes     |       1 |
```

```
| utf8mb4_vietnamese_ci | utf8mb4 | 247 |         | Yes     |       1 |
+--
    ↪ ----------------------+---------+-----+---------+---------+---------+
    ↪
219 rows in set (0.00 sec)
```

4.1.5.64.3 MySQL compatibility

The usage of this statement is understood to be fully compatible with MySQL. However, charsets in TiDB might have different default collations compared with MySQL. For details, refer to Compatibility with MySQL. Any other compatibility differences should be reported via an issue on GitHub.

4.1.5.64.4 See also

- SHOW CHARACTER SET

4.1.5.65 SHOW [FULL] COLUMNS FROM

The statement `SHOW [FULL] COLUMNS FROM <table_name>` describes the columns of a table or view in a useful tabular format. The optional keyword `FULL` displays the privileges the current user has to that column, and the `comment` from the table definition.

The statements `SHOW [FULL] FIELDS FROM <table_name>`, `DESC <table_name>`, `DESCRIBE <table_name>`, and `EXPLAIN <table_name>` are aliases of this statement.

> **Note:**
>
> `DESC TABLE <table_name>`, `DESCRIBE TABLE <table_name>`, and `EXPLAIN TABLE <table_name>` are not equivalent to the above statements. They are aliases of `DESC SELECT * FROM <table_name>`.

4.1.5.65.1 Synopsis

ShowStmt:

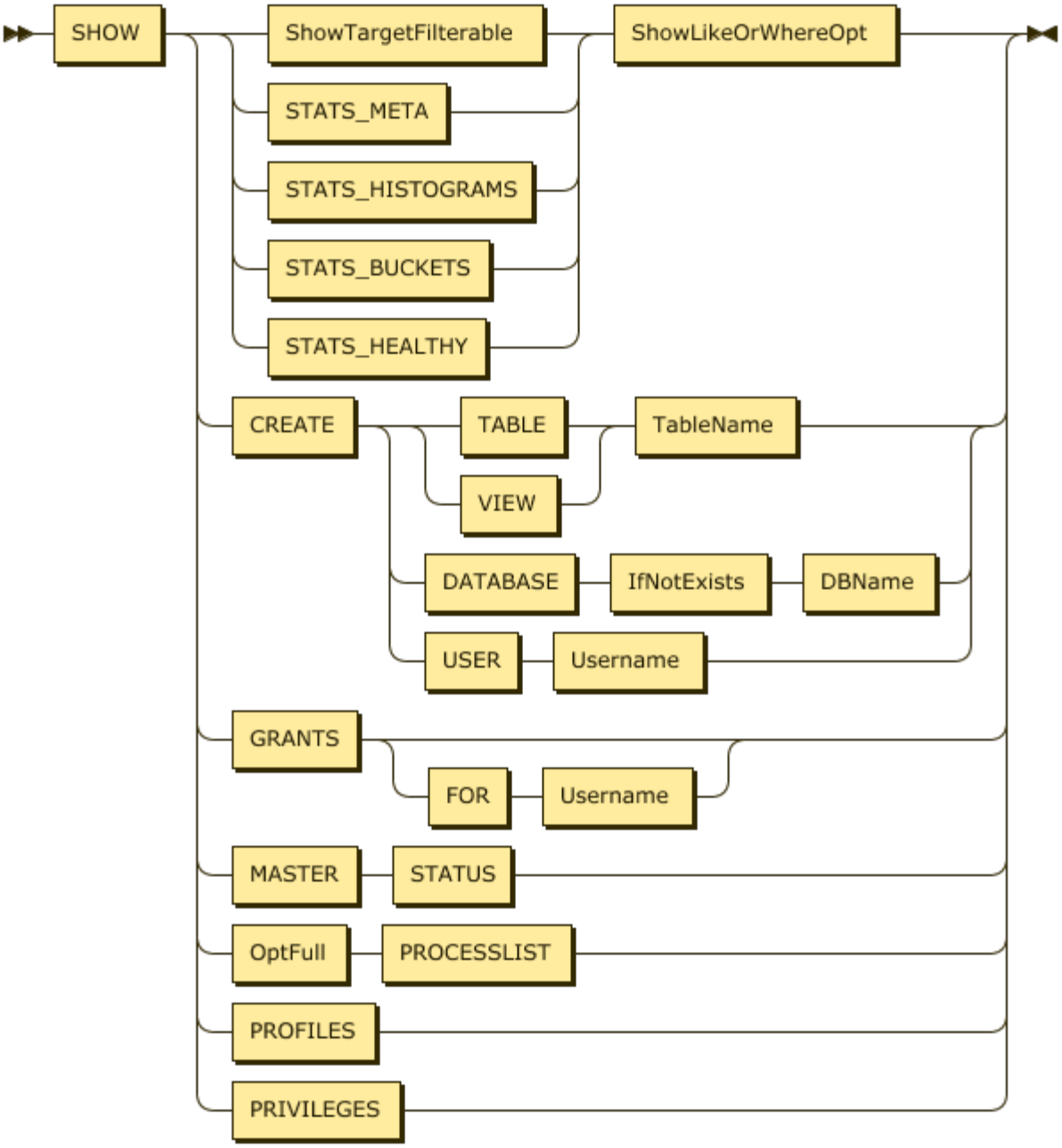

Figure 216: ShowStmt

ShowTargetFilterable:

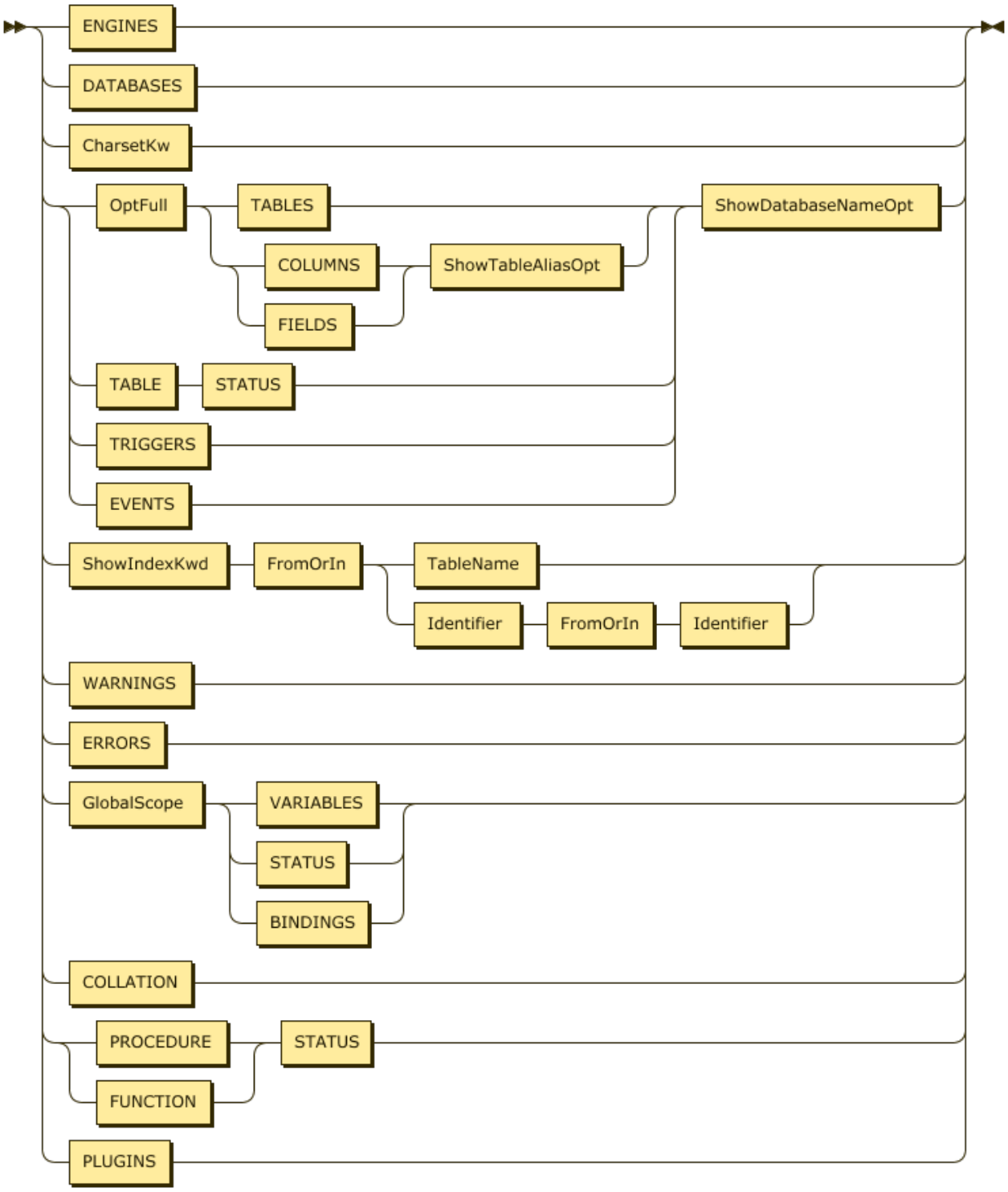

Figure 217: ShowTargetFilterable

OptFull:

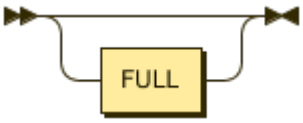

Figure 218: OptFull

4.1.5.65.2 Examples

```
mysql> create view v1 as select 1;
Query OK, 0 rows affected (0.11 sec)

mysql> show columns from v1;
+-------+-----------+------+------+---------+-------+
| Field | Type      | Null | Key | Default | Extra |
+-------+-----------+------+------+---------+-------+
| 1     | bigint(1) | YES |     | NULL    |       |
+-------+-----------+------+------+---------+-------+
1 row in set (0.00 sec)

mysql> desc v1;
+-------+-----------+------+------+---------+-------+
| Field | Type      | Null | Key | Default | Extra |
+-------+-----------+------+------+---------+-------+
| 1     | bigint(1) | YES |     | NULL    |       |
+-------+-----------+------+------+---------+-------+
1 row in set (0.00 sec)

mysql> describe v1;
+-------+-----------+------+------+---------+-------+
| Field | Type      | Null | Key | Default | Extra |
+-------+-----------+------+------+---------+-------+
| 1     | bigint(1) | YES |     | NULL    |       |
+-------+-----------+------+------+---------+-------+
1 row in set (0.00 sec)

mysql> explain v1;
+-------+-----------+------+------+---------+-------+
| Field | Type      | Null | Key | Default | Extra |
+-------+-----------+------+------+---------+-------+
| 1     | bigint(1) | YES |     | NULL    |       |
+-------+-----------+------+------+---------+-------+
1 row in set (0.00 sec)

mysql> show fields from v1;
+-------+-----------+------+------+---------+-------+
```

```
| Field | Type    | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| 1     | bigint(1) | YES |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
1 row in set (0.00 sec)

mysql> show full columns from v1;
+--
    ↪ -----+----------+----------+------+-----+---------+-------+-----------------
    ↪
| Field | Type    | Collation | Null | Key | Default | Extra | Privileges
    ↪              | Comment |
+--
    ↪ -----+----------+----------+------+-----+---------+-------+-----------------
    ↪
| 1     | bigint(1) | NULL    | YES |     | NULL    |       | select,insert,
    ↪ update,references |    |
+--
    ↪ -----+----------+----------+------+-----+---------+-------+-----------------
    ↪
1 row in set (0.00 sec)

mysql> show full columns from mysql.user;
+--
    ↪ --------------------+--------------+------------+------+-----+---------+----
    ↪
| Field                  | Type         | Collation  | Null | Key | Default |
    ↪ Extra | Privileges                | Comment |
+--
    ↪ --------------------+--------------+------------+------+-----+---------+----
    ↪
| Host                   | char(64)     | utf8mb4_bin | NO | PRI | NULL    |
    ↪ | select,insert,update,references |    |
| User                   | char(32)     | utf8mb4_bin | NO | PRI | NULL    |
    ↪ | select,insert,update,references |    |
| Password               | char(41)     | utf8mb4_bin | YES |    | NULL    |
    ↪ | select,insert,update,references |    |
| Select_priv            | enum('N','Y') | utf8mb4_bin | NO |   | N       |
    ↪ | select,insert,update,references |    |
| Insert_priv            | enum('N','Y') | utf8mb4_bin | NO |   | N       |
    ↪ | select,insert,update,references |    |
| Update_priv            | enum('N','Y') | utf8mb4_bin | NO |   | N       |
    ↪ | select,insert,update,references |    |
| Delete_priv            | enum('N','Y') | utf8mb4_bin | NO |   | N       |
    ↪ | select,insert,update,references |    |
```

```
| Create_priv         | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Drop_priv           | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Process_priv        | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Grant_priv          | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| References_priv     | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Alter_priv          | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Show_db_priv        | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Super_priv          | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Create_tmp_table_priv | enum('N','Y') | utf8mb4_bin | NO | | N    |
   ↪ | select,insert,update,references |    |
| Lock_tables_priv    | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Execute_priv        | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Create_view_priv    | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Show_view_priv      | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Create_routine_priv | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Alter_routine_priv  | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Index_priv          | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Create_user_priv    | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Event_priv          | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Trigger_priv        | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Create_role_priv    | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Drop_role_priv      | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
| Account_locked      | enum('N','Y') | utf8mb4_bin | NO |   | N      |
   ↪ | select,insert,update,references |    |
+--
```

```
    ↪ --------------------+--------------+------------+------+------+----------+----
    ↪
29 rows in set (0.00 sec)
```

4.1.5.65.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.65.4 See also

- SHOW CREATE TABLE

4.1.5.66 SHOW CREATE TABLE

This statement shows the exact statement to recreate an existing table using SQL.

4.1.5.66.1 Synopsis

ShowStmt:

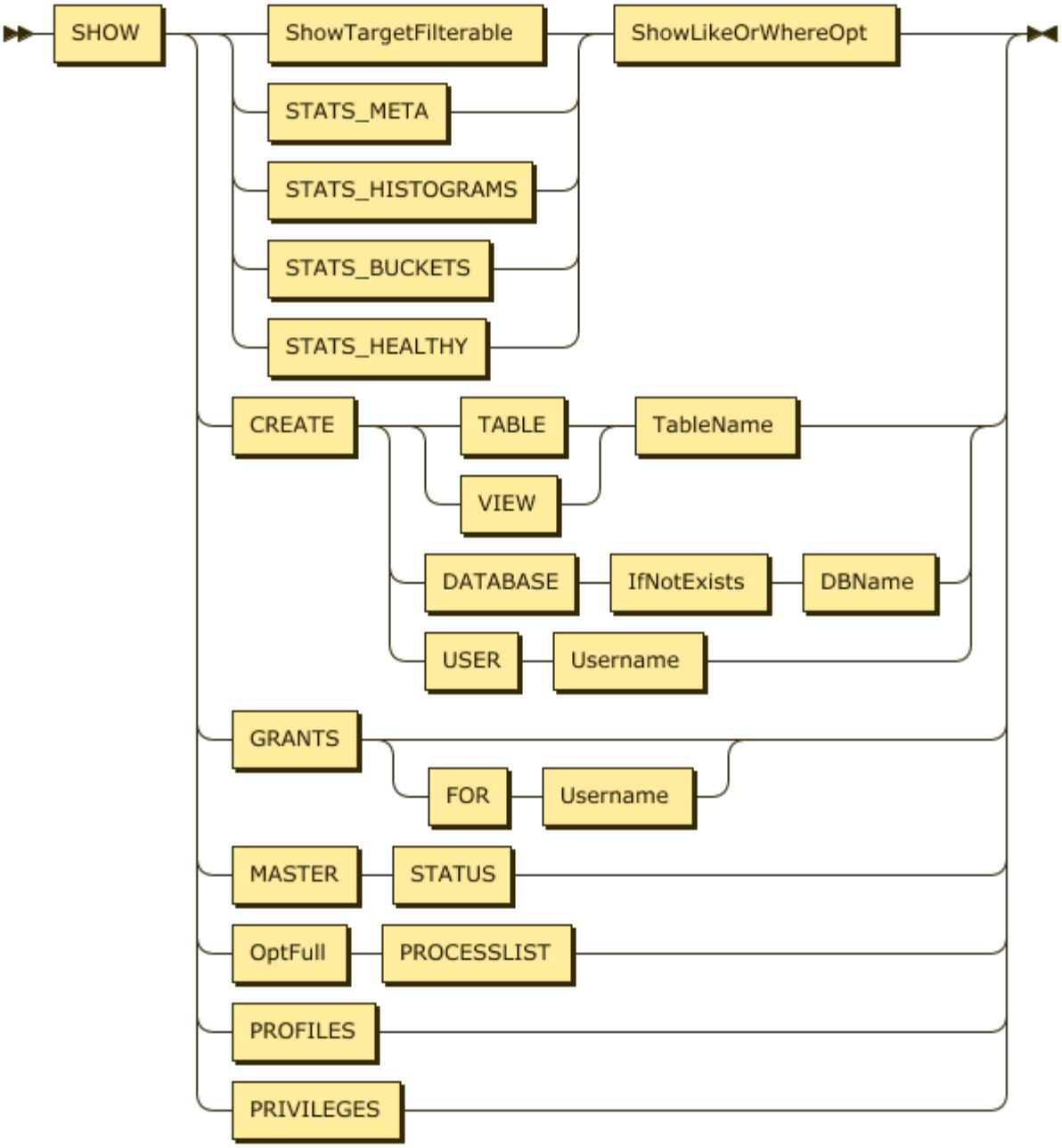

Figure 219: ShowStmt

TableName:

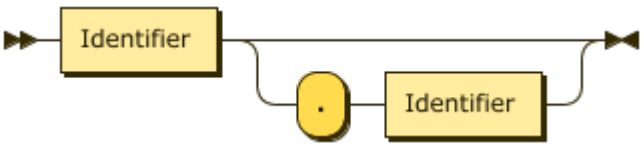

Figure 220: TableName

4.1.5.66.2 Examples

```
mysql> CREATE TABLE t1 (a INT);
Query OK, 0 rows affected (0.12 sec)

mysql> SHOW CREATE TABLE t1;
+--
    ↪ -----+-------------------------------------------------------------------------
    ↪
| Table | Create Table
    ↪
    ↪ |
+--
    ↪ -----+-------------------------------------------------------------------------
    ↪
| t1    | CREATE TABLE `t1` (
  `a` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin |
+--
    ↪ -----+-------------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)
```

4.1.5.66.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.66.4 See also

- CREATE TABLE
- DROP TABLE
- SHOW TABLES
- SHOW COLUMNS FROM

4.1.5.67 SHOW CREATE USER

This statement shows how to re-create a user using the `CREATE USER` syntax.

4.1.5.67.1 Synopsis

ShowStmt:

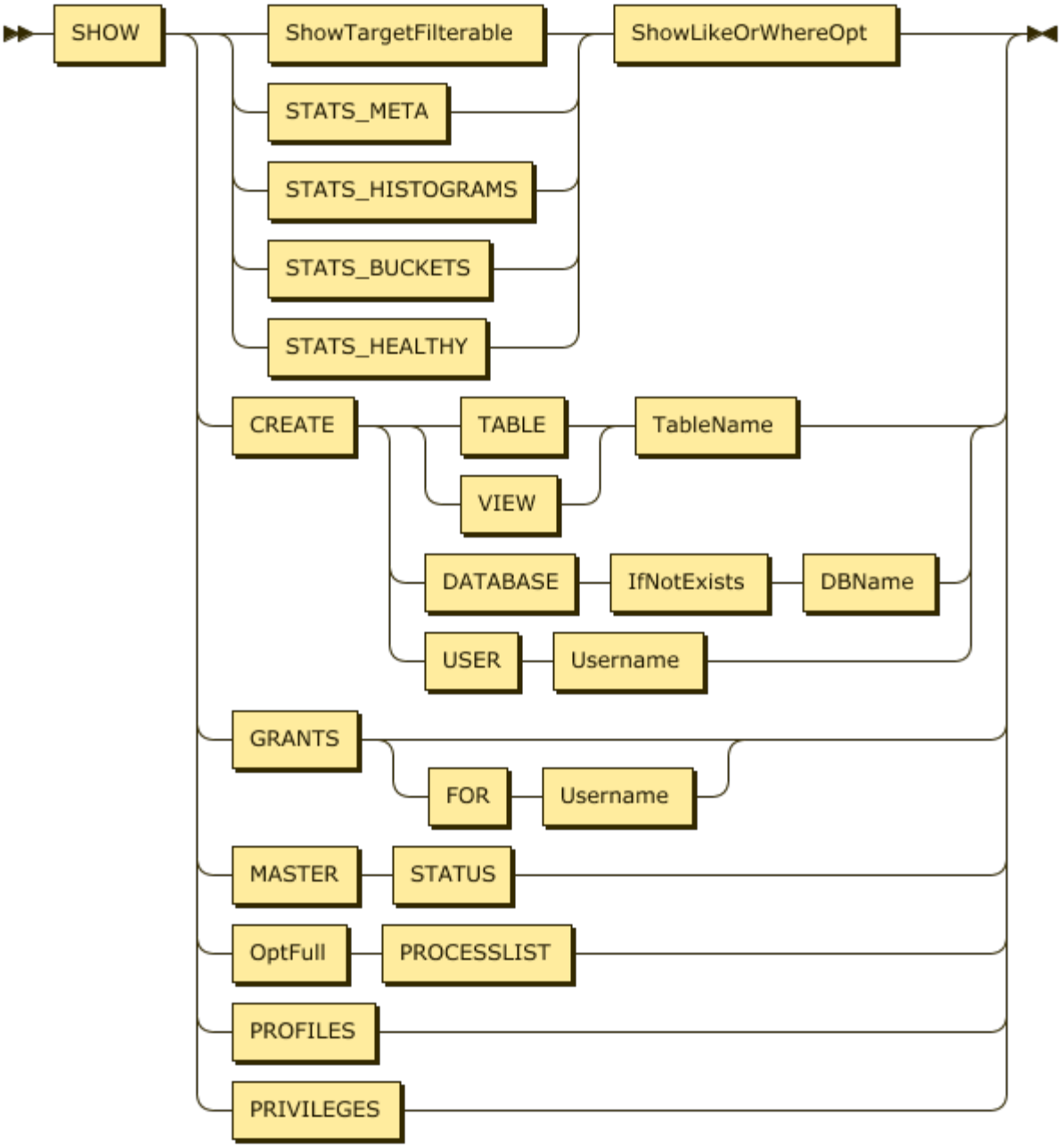

Figure 221: ShowStmt

Username:

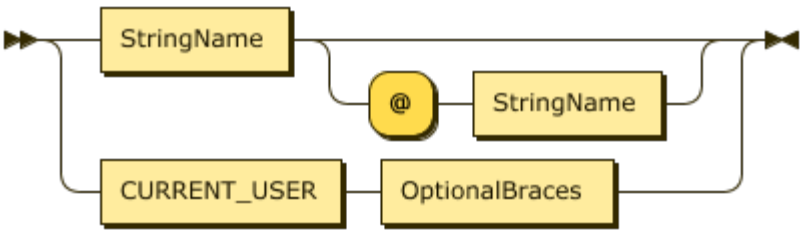

Figure 222: Username

4.1.5.67.2 Examples

```
mysql> SHOW CREATE USER 'root';
+--
    ↪ ------------------------------------------------------------------------------
    ↪
| CREATE USER for root@%
    ↪
    ↪ |
+--
    ↪ ------------------------------------------------------------------------------
    ↪
| CREATE USER 'root'@'%' IDENTIFIED WITH 'mysql_native_password' AS ''
    ↪ REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK |
+--
    ↪ ------------------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)

mysql> SHOW GRANTS FOR 'root';
+-----------------------------------------+
| Grants for root@%                       |
+-----------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' |
+-----------------------------------------+
1 row in set (0.00 sec)
```

4.1.5.67.3 MySQL compatibility

- The output of `SHOW CREATE USER` is designed to match MySQL, but several of the `CREATE` options are not yet supported by TiDB. Not yet supported options will be parsed but ignored. See [security compatibility] for more details.

4.1.5.67.4 See also

- CREATE USER
- SHOW GRANTS
- DROP USER

4.1.5.68 SHOW DATABASES

This statement shows a list of databases that the current user has privileges to. Databases which the current user does not have access to will appear hidden from the list. The `information_schema` database always appears first in the list of databases.

`SHOW SCHEMAS` is an alias of this statement.

4.1.5.68.1 Synopsis

ShowStmt:

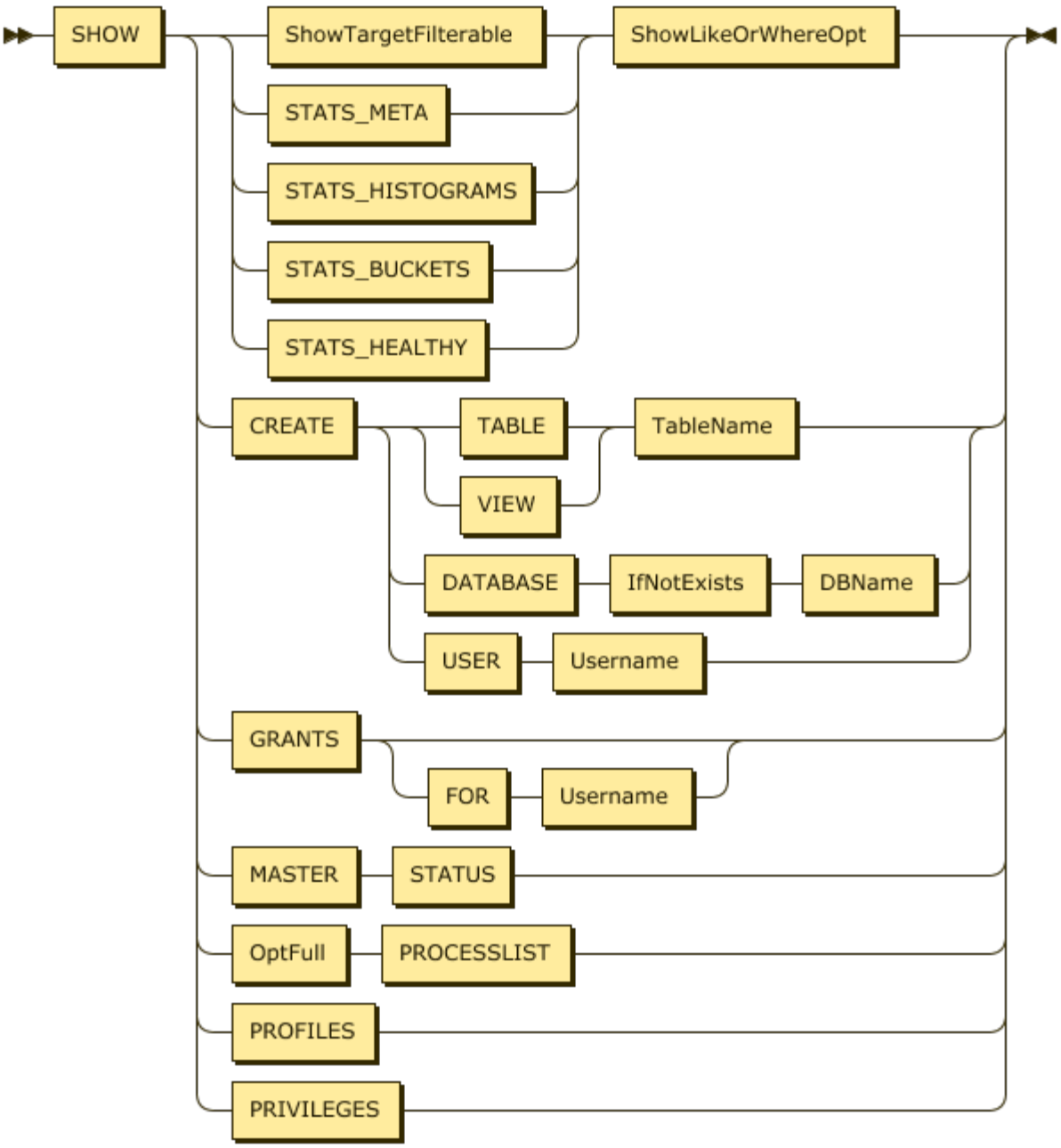

Figure 223: ShowStmt

ShowTargetFilterable:

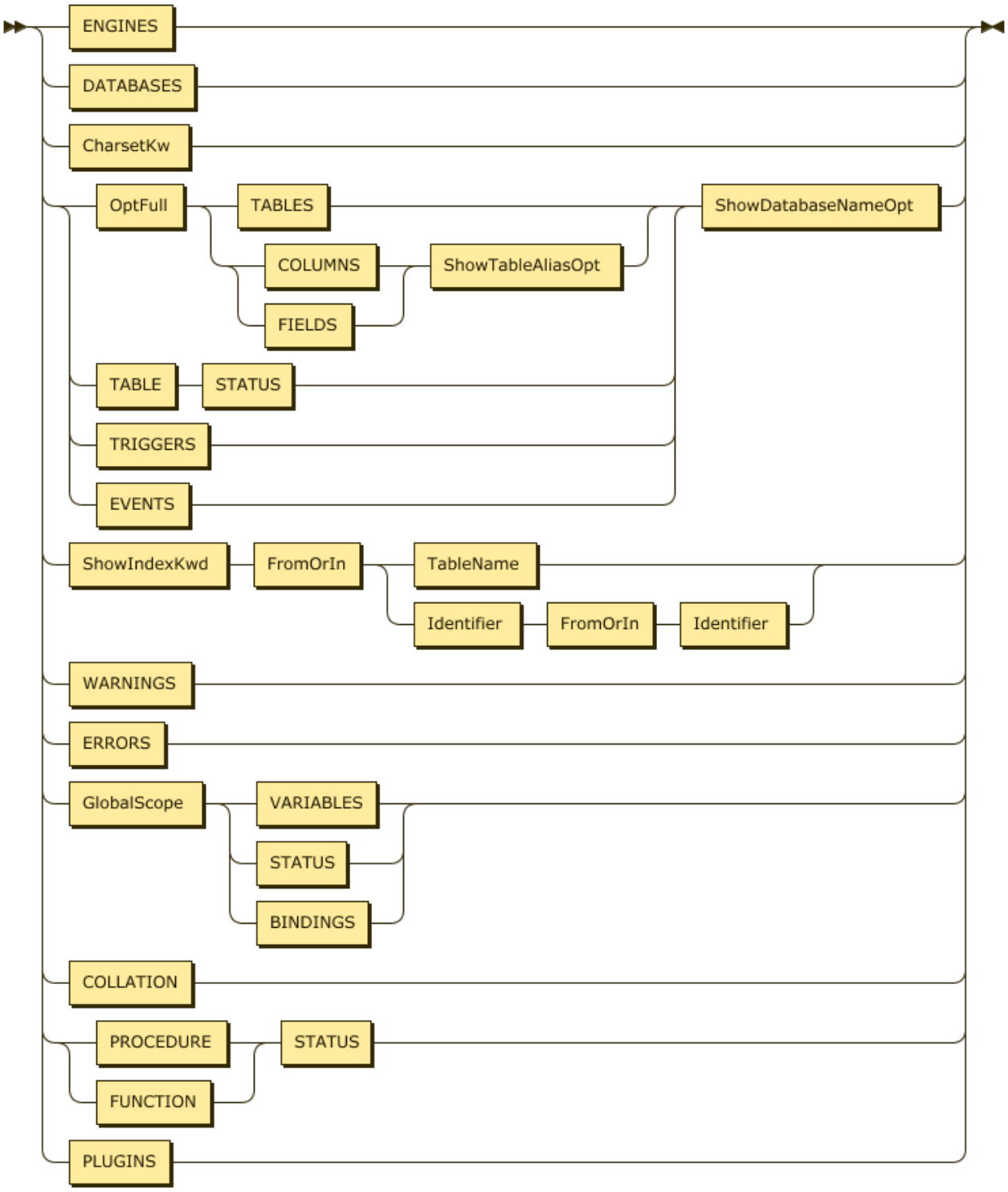

Figure 224: ShowTargetFilterable

4.1.5.68.2 Examples

```
mysql> SHOW DATABASES;
```

```
+--------------------+
| Database           |
+--------------------+
| INFORMATION_SCHEMA |
| PERFORMANCE_SCHEMA |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql> CREATE DATABASE mynewdb;
Query OK, 0 rows affected (0.10 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| INFORMATION_SCHEMA |
| PERFORMANCE_SCHEMA |
| mynewdb            |
| mysql              |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

4.1.5.68.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.68.4 See also

- SHOW SCHEMAS
- DROP DATABASE
- CREATE DATABASE

4.1.5.69 SHOW ENGINES

This statement is included only for compatibility with MySQL.

4.1.5.69.1 Synopsis

```
SHOW ENGINES
```

4.1.5.69.2 Examples

```
mysql> SHOW ENGINES;
+--
    ↪ ------+---------+----------------------------------------------------------+--
    ↪
| Engine | Support | Comment                                                  |
    ↪ Transactions | XA | Savepoints |
+--
    ↪ ------+---------+----------------------------------------------------------+--
    ↪
| InnoDB | DEFAULT | Supports transactions, row-level locking, and foreign
    ↪ keys | YES | YES | YES   |
+--
    ↪ ------+---------+----------------------------------------------------------+--
    ↪
1 row in set (0.00 sec)
```

4.1.5.69.3 MySQL compatibility

- This statement will always only return InnoDB as the supported engine. Internally, TiDB will typically use TiKV as the storage engine.

4.1.5.70 SHOW ERRORS

This statement shows error(s) from previously executed statements. The error buffer is cleared as soon as a statement executes successfully. In which case, `SHOW ERRORS` will return an empty set.

The behavior of which statements generate errors vs. warnings is highly influenced by the current `sql_mode`.

4.1.5.70.1 Synopsis

ShowStmt:

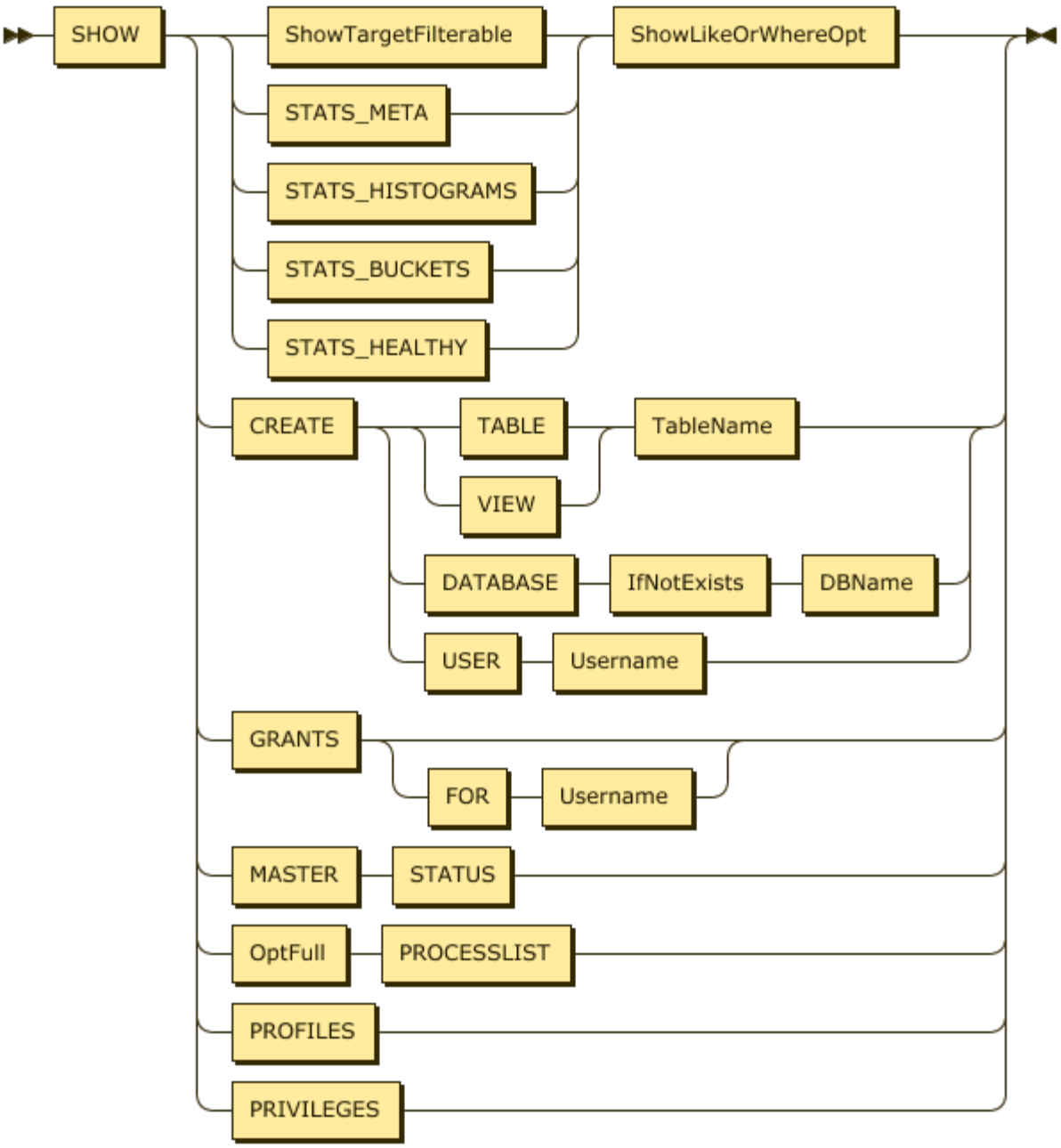

Figure 225: ShowStmt

ShowTargetFilterable:

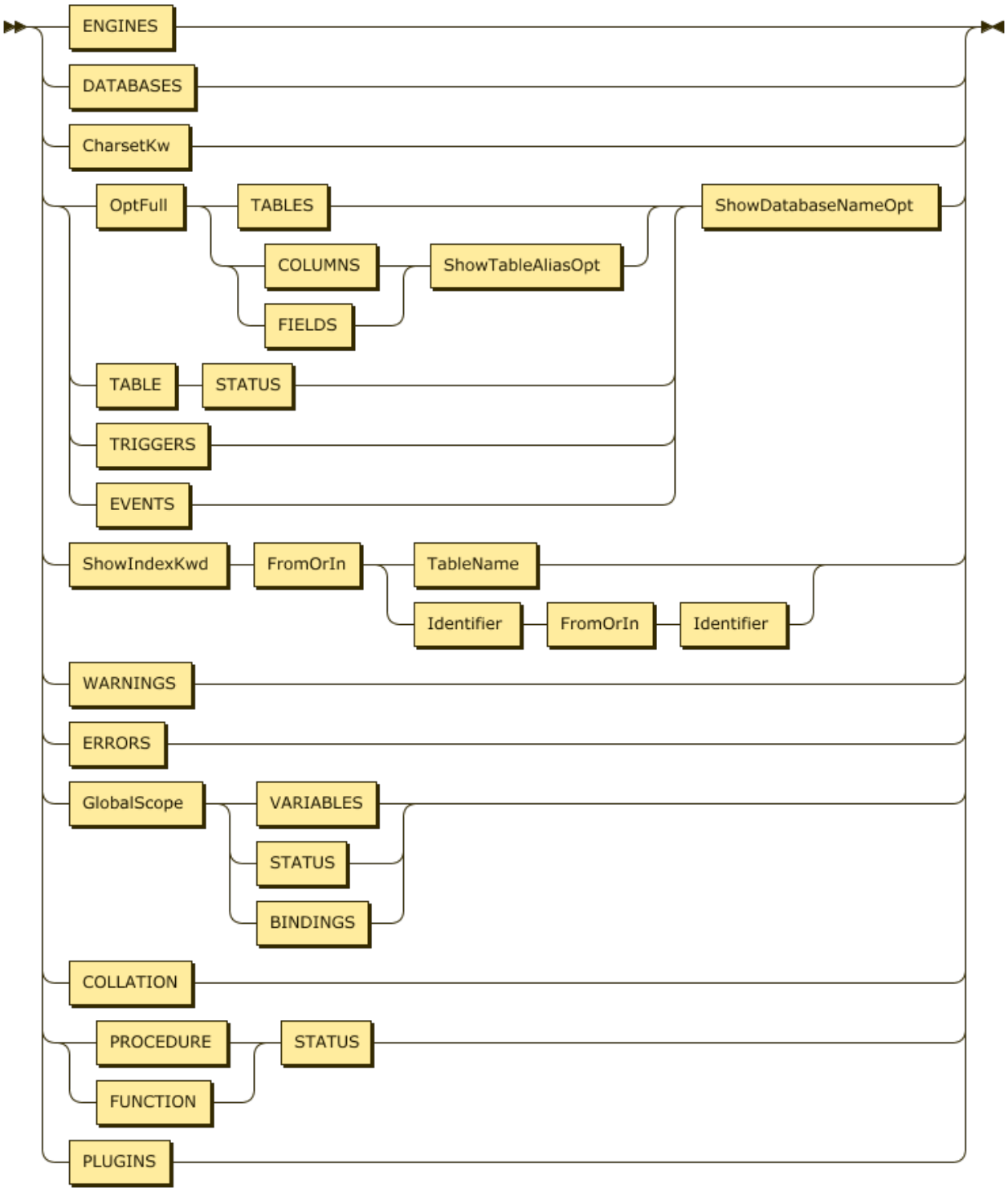

Figure 226: ShowTargetFilterable

4.1.5.70.2 Examples

```
mysql> select invalid;
```

```
ERROR 1054 (42S22): Unknown column 'invalid' in 'field list'
mysql> create invalid;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
    ↪ that corresponds to your TiDB version for the right syntax to use
    ↪ line 1 column 14 near "invalid"
mysql> SHOW ERRORS;
+--
    ↪ -----+------+-----------------------------------------------------------------
    ↪
| Level | Code | Message
    ↪
    ↪ |
+--
    ↪ -----+------+-----------------------------------------------------------------
    ↪
| Error | 1054 | Unknown column 'invalid' in 'field list'
    ↪
    ↪ |
| Error | 1064 | You have an error in your SQL syntax; check the manual
    ↪ that corresponds to your TiDB version for the right syntax to use
    ↪ line 1 column 14 near "invalid" |
+--
    ↪ -----+------+-----------------------------------------------------------------
    ↪
2 rows in set (0.00 sec)

mysql> CREATE invalid2;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual
    ↪ that corresponds to your TiDB version for the right syntax to use
    ↪ line 1 column 15 near "invalid2"
mysql> SELECT 1;
+------+
| 1    |
+------+
|    1 |
+------+
1 row in set (0.00 sec)

mysql> SHOW ERRORS;
Empty set (0.00 sec)
```

4.1.5.70.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility

differences should be reported via an issue on GitHub.

4.1.5.70.4 See also

- SHOW WARNINGS

4.1.5.71 SHOW [FULL] FIELDS FROM

This statement is an alias to `SHOW [FULL] COLUMNS FROM`. It is included for compatibility with MySQL.

4.1.5.72 SHOW GRANTS

This statement shows a list of privileges associated with a user. As in MySQL, the `USAGE` privileges denotes the ability to login to TiDB.

4.1.5.72.1 Synopsis

ShowStmt:

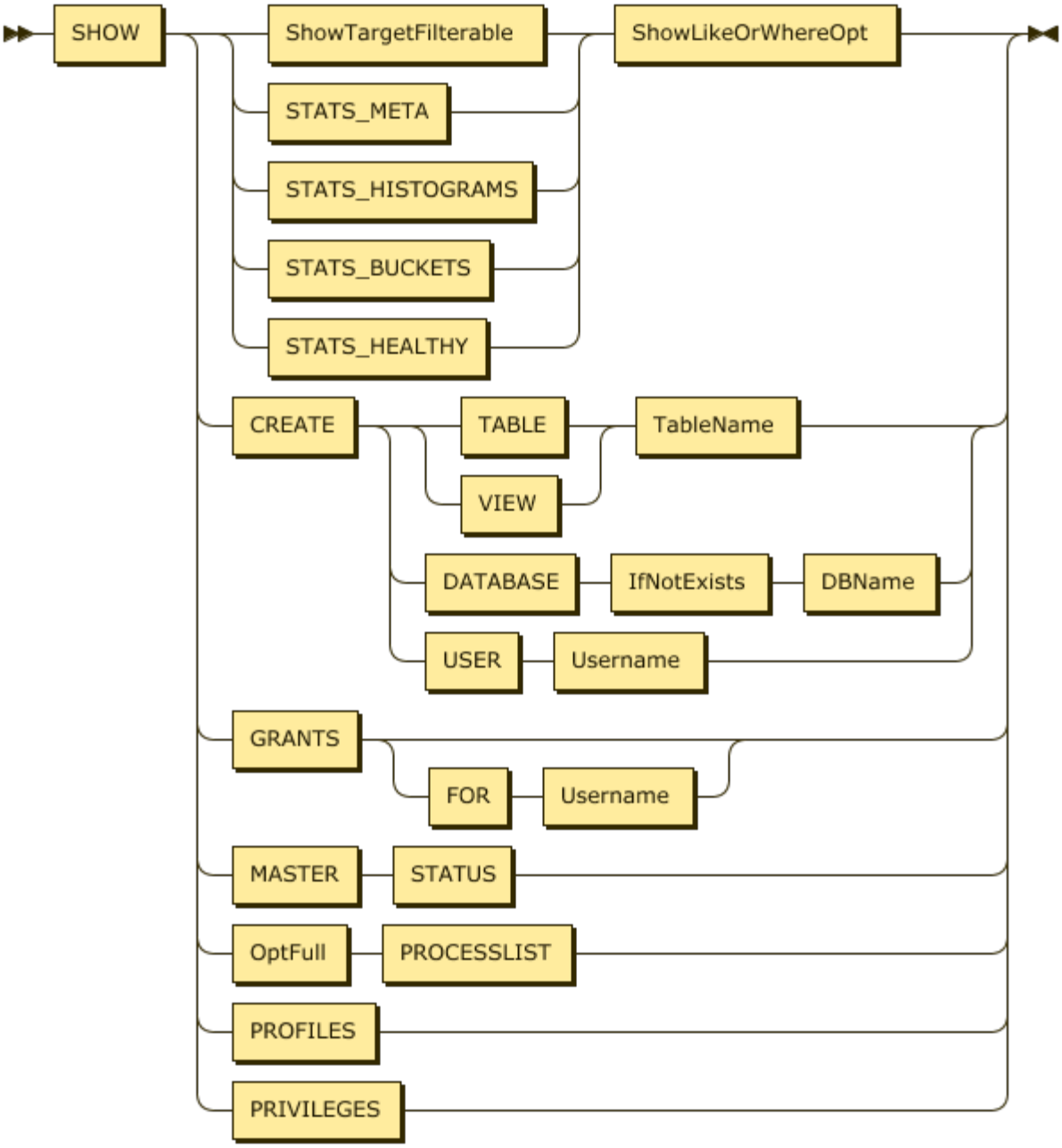

Figure 227: ShowStmt

Username:

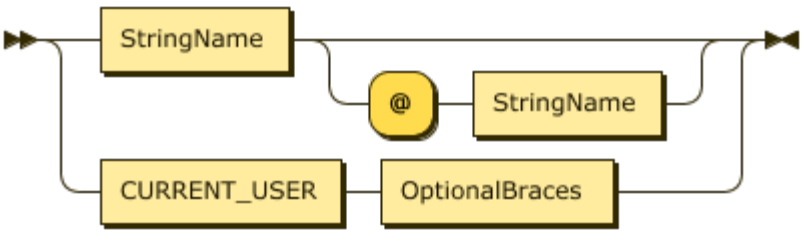

Figure 228: Username

4.1.5.72.2 Examples

```
mysql> SHOW GRANTS;
+----------------------------------------+
| Grants for User                        |
+----------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' |
+----------------------------------------+
1 row in set (0.00 sec)

mysql> SHOW GRANTS FOR 'u1';
ERROR 1141 (42000): There is no such grant defined for user 'u1' on host '%
    ↪ '
mysql> CREATE USER u1;
Query OK, 1 row affected (0.04 sec)

mysql> GRANT SELECT ON test.* TO u1;
Query OK, 0 rows affected (0.04 sec)

mysql> SHOW GRANTS FOR u1;
+-----------------------------------+
| Grants for u1@%                   |
+-----------------------------------+
| GRANT USAGE ON *.* TO 'u1'@'%' |
| GRANT Select ON test.* TO 'u1'@'%' |
+-----------------------------------+
2 rows in set (0.00 sec)
```

4.1.5.72.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.72.4 See also

- SHOW CREATE USER

- GRANT

4.1.5.73 SHOW INDEXES [FROM|IN]

The statement `SHOW INDEXES [FROM|IN]` lists the indexes on a specified table. The statements `SHOW INDEX [FROM|IN]`, `SHOW KEYS [FROM|IN]` are aliases of this statement, and included for compatibility with MySQL.

4.1.5.73.1 Synopsis

ShowStmt:

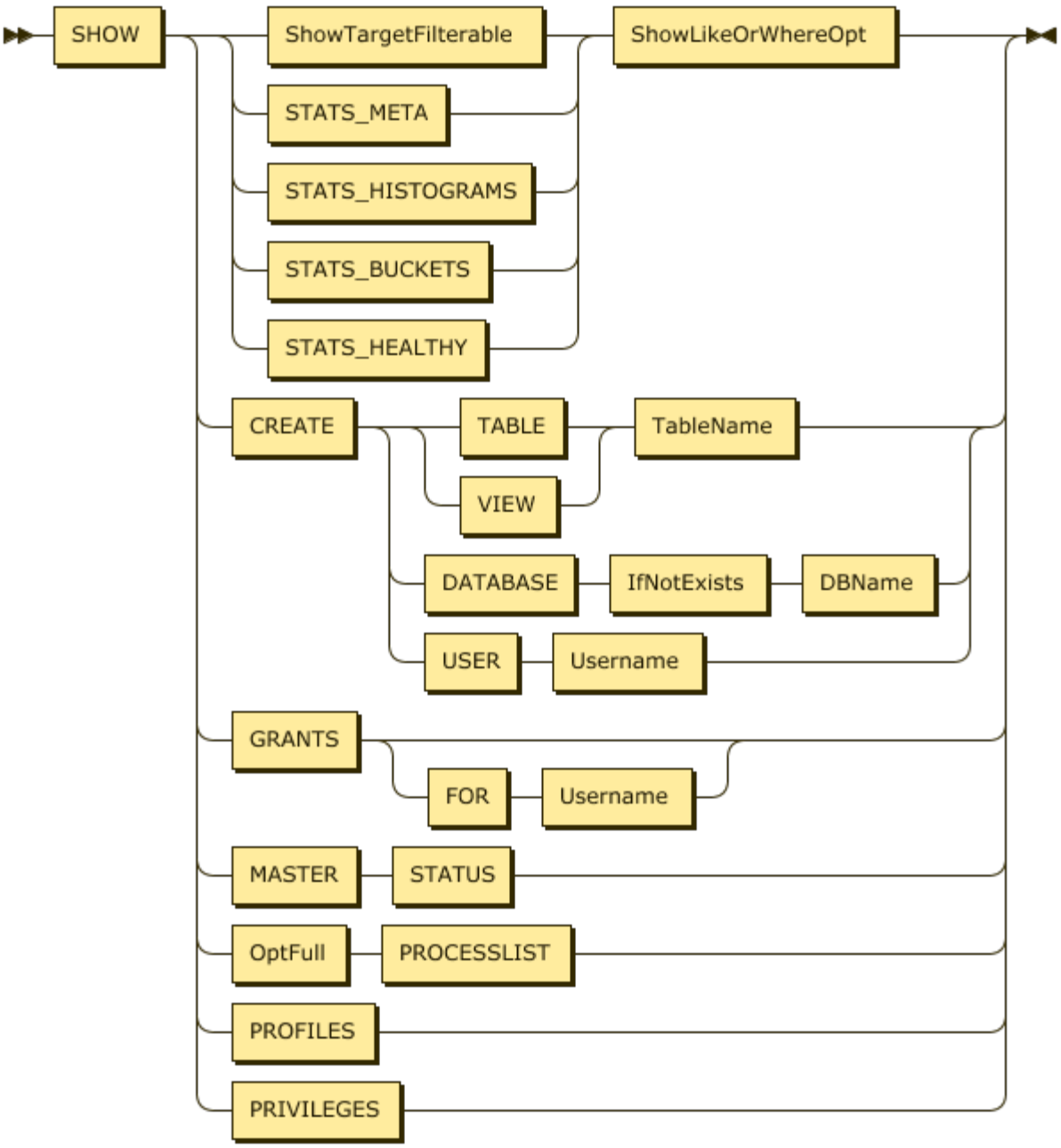

Figure 229: ShowStmt

ShowTargetFilterable:

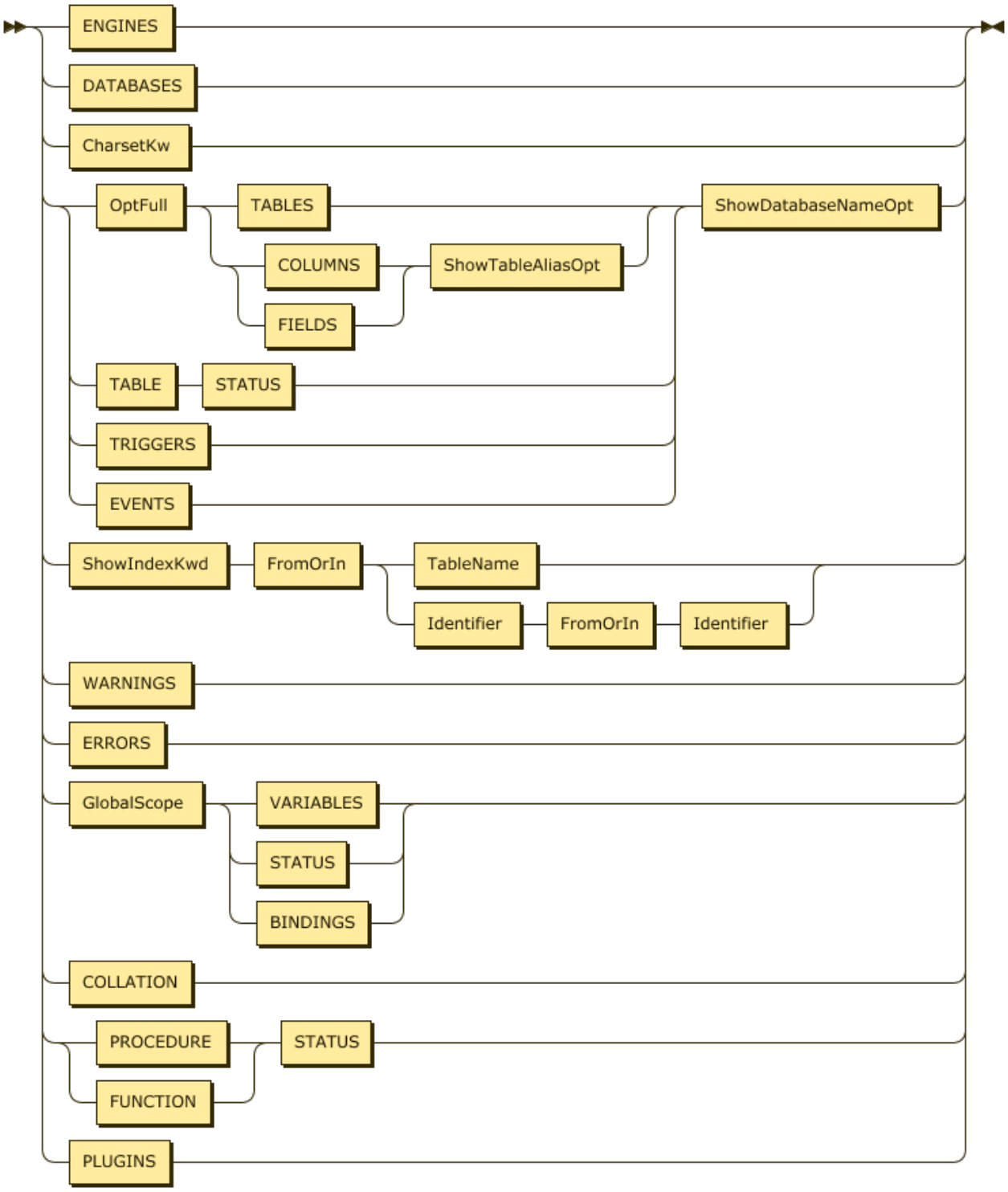

Figure 230: ShowTargetFilterable

ShowIndexKwd:

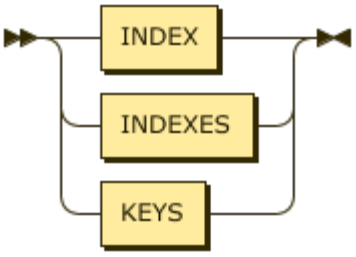

Figure 231: ShowIndexKwd

FromOrIn:

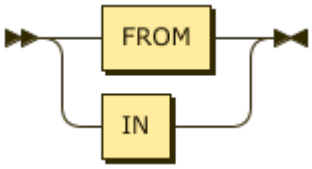

Figure 232: FromOrIn

TableName:

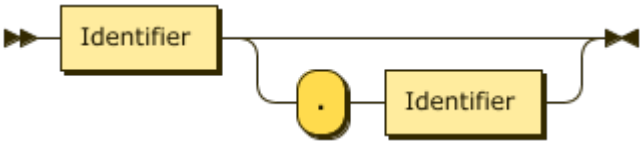

Figure 233: TableName

4.1.5.73.2 Examples

```
mysql> CREATE TABLE t1 (id int not null primary key AUTO_INCREMENT, col1
    ↪ INT, INDEX(col1));
Query OK, 0 rows affected (0.12 sec)

mysql> SHOW INDEXES FROM t1;
+--
    ↪ -----+------------+----------+--------------+-------------+-----------+---------
    ↪
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation |
    ↪ Cardinality | Sub_part | Packed | Null | Index_type | Comment |
    ↪ Index_comment |
+--
    ↪ -----+------------+----------+--------------+-------------+-----------+---------
    ↪
| t1    |          0 | PRIMARY |            1 | id          | A         |           0
    ↪  |     NULL | NULL |      | BTREE      |             |               |
```

```
| t1    |          1 | col1     |            1 | col1       | A         |           0
    ↪  |    NULL | NULL | YES | BTREE      |         |               |
+--
    ↪ -----+------------+----------+-------------+------------+-----------+---------
    ↪
2 rows in set (0.00 sec)

mysql> SHOW INDEX FROM t1;
+--
    ↪ -----+------------+----------+-------------+------------+-----------+---------
    ↪
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation |
    ↪ Cardinality | Sub_part | Packed | Null | Index_type | Comment |
    ↪ Index_comment |
+--
    ↪ -----+------------+----------+-------------+------------+-----------+---------
    ↪
| t1    |          0 | PRIMARY  |            1 | id         | A         |           0
    ↪  |    NULL | NULL |      | BTREE      |         |               |
| t1    |          1 | col1     |            1 | col1       | A         |           0
    ↪  |    NULL | NULL | YES | BTREE      |         |               |
+--
    ↪ -----+------------+----------+-------------+------------+-----------+---------
    ↪
2 rows in set (0.00 sec)

mysql> SHOW KEYS FROM t1;
+--
    ↪ -----+------------+----------+-------------+------------+-----------+---------
    ↪
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation |
    ↪ Cardinality | Sub_part | Packed | Null | Index_type | Comment |
    ↪ Index_comment |
+--
    ↪ -----+------------+----------+-------------+------------+-----------+---------
    ↪
| t1    |          0 | PRIMARY  |            1 | id         | A         |           0
    ↪  |    NULL | NULL |      | BTREE      |         |               |
| t1    |          1 | col1     |            1 | col1       | A         |           0
    ↪  |    NULL | NULL | YES | BTREE      |         |               |
+--
    ↪ -----+------------+----------+-------------+------------+-----------+---------
    ↪
2 rows in set (0.00 sec)
```

4.1.5.73.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.73.4 See also

- SHOW CREATE TABLE
- DROP INDEX
- CREATE INDEX

4.1.5.74 SHOW INDEX [FROM|IN]

This statement is an alias to `SHOW INDEXES [FROM|IN]`. It is included for compatibility with MySQL.

4.1.5.75 SHOW KEYS [FROM|IN]

This statement is an alias to `SHOW INDEXES [FROM|IN]`. It is included for compatibility with MySQL.

4.1.5.76 SHOW PRIVILEGES

This statement shows a list of assignable privileges in TiDB. It is a static list, and does not reflect the privileges of the current user.

4.1.5.76.1 Synopsis

ShowStmt:

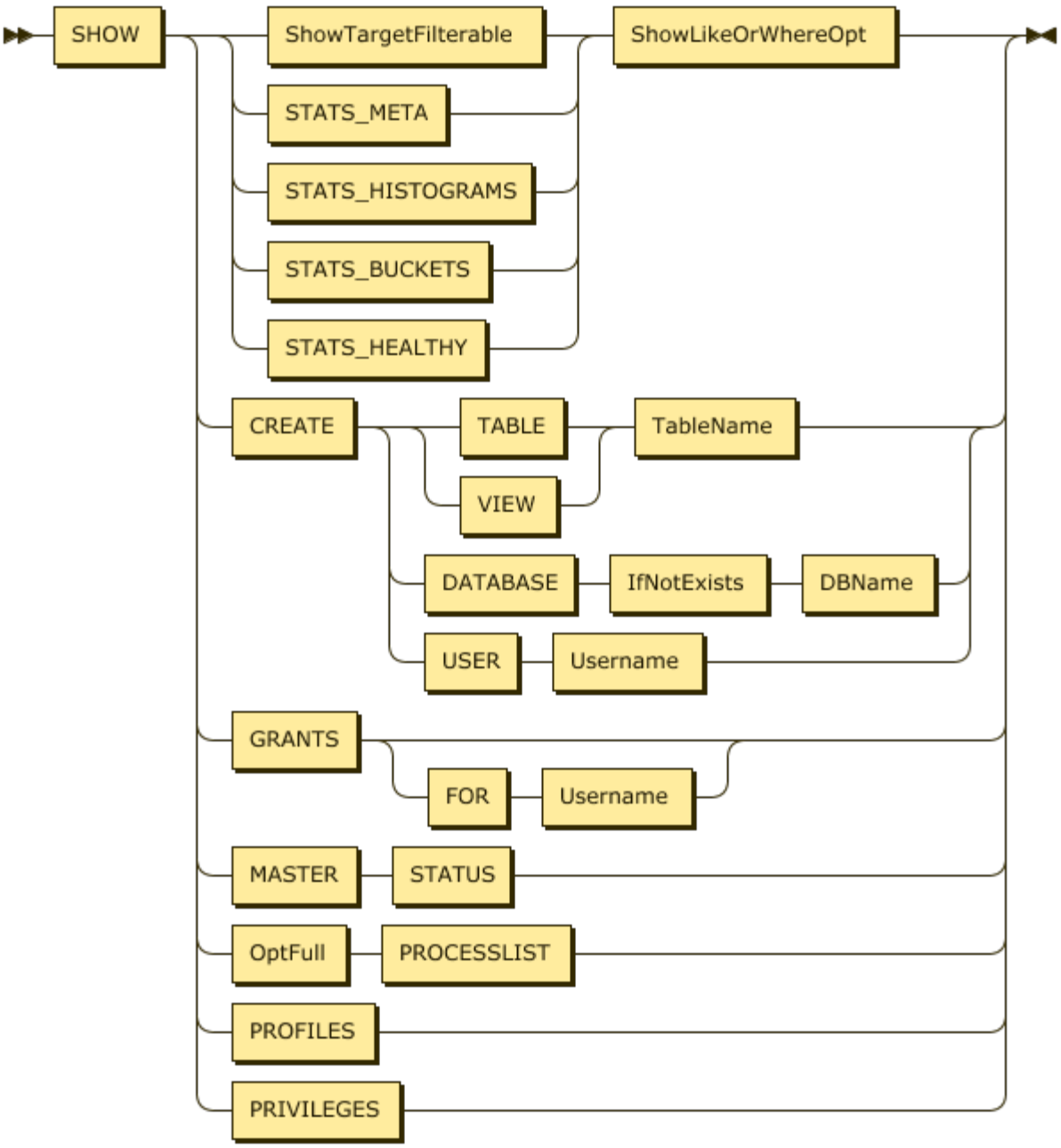

Figure 234: ShowStmt

4.1.5.76.2 Examples

```
mysql> show privileges;
+--
    ↪ ----------------------+-----------------------------------------+---------------
    ↪
| Privilege             | Context                                 | Comment
    ↪                                                 |
+--
```

```
   ↪ ---------------------+----------------------------------------+---------------
   ↪
| Alter                | Tables                                 | To alter the
   ↪ table                                |
| Alter                | Tables                                 | To alter the
   ↪ table                                |
| Alter routine        | Functions,Procedures                   | To alter or drop
   ↪  stored functions/procedures   |
| Create               | Databases,Tables,Indexes               | To create new
   ↪ databases and tables                 |
| Create routine       | Databases                              | To use CREATE
   ↪ FUNCTION/PROCEDURE                   |
| Create temporary tables | Databases                           | To use CREATE
   ↪ TEMPORARY TABLE                      |
| Create view          | Tables                                 | To create new
   ↪ views                                |
| Create user          | Server Admin                           | To create new
   ↪ users                                |
| Delete               | Tables                                 | To delete
   ↪ existing rows                            |
| Drop                 | Databases,Tables                       | To drop
   ↪ databases, tables, and views           |
| Event                | Server Admin                           | To create, alter
   ↪ , drop and execute events      |
| Execute              | Functions,Procedures                   | To execute
   ↪ stored routines                          |
| File                 | File access on server                  | To read and
   ↪ write files on the server          |
| Grant option         | Databases,Tables,Functions,Procedures | To give to
   ↪ other users those privileges you possess |
| Index                | Tables                                 | To create or
   ↪ drop indexes                         |
| Insert               | Tables                                 | To insert data
   ↪ into tables                        |
| Lock tables          | Databases                              | To use LOCK
   ↪ TABLES (together with SELECT privilege) |
| Process              | Server Admin                           | To view the
   ↪ plain text of currently executing queries |
| Proxy                | Server Admin                           | To make proxy
   ↪ user possible                        |
| References           | Databases,Tables                       | To have
   ↪ references on tables                     |
| Reload               | Server Admin                           | To reload or
   ↪ refresh tables, logs and privileges |
| Replication client   | Server Admin                           | To ask where the
```

```
   ↪  slave or master servers are   |
| Replication slave    | Server Admin                    | To read binary
   ↪ log events from the master      |
| Select               | Tables                          | To retrieve rows
   ↪  from table                    |
| Show databases       | Server Admin                    | To see all
   ↪ databases with SHOW DATABASES       |
| Show view            | Tables                          | To see views
   ↪ with SHOW CREATE VIEW             |
| Shutdown             | Server Admin                    | To shut down the
   ↪  server                        |
| Super                | Server Admin                    | To use KILL
   ↪ thread, SET GLOBAL, CHANGE MASTER, etc. |
| Trigger              | Tables                          | To use triggers
   ↪                                |
| Create tablespace    | Server Admin                    | To create/alter/
   ↪ drop tablespaces               |
| Update               | Tables                          | To update
   ↪ existing rows                       |
| Usage                | Server Admin                    | No privileges -
   ↪ allow connect only             |
+--
   ↪ ----------------------+----------------------------------------+----------------
   ↪
32 rows in set (0.00 sec)
```

4.1.5.76.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.76.4 See also

- SHOW GRANTS
- GRANT <privileges>

4.1.5.77 SHOW [FULL] PROCESSLIST

This statement lists the current sessions connected to the same TiDB server. The `Info` column contains the query text, which will be truncated unless the optional keyword `FULL` is specified.

4.1.5.77.1 Synopsis

ShowStmt:

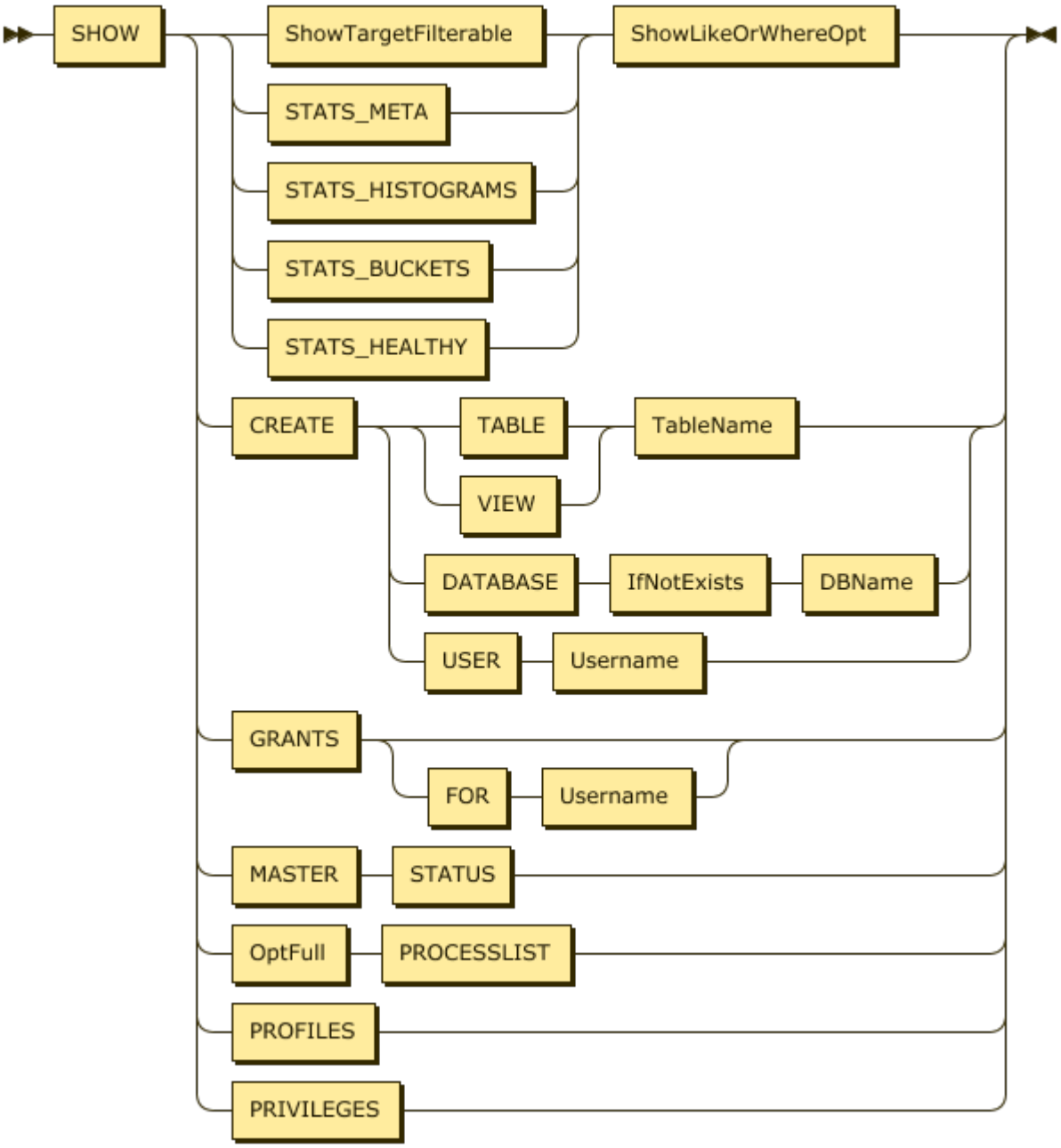

Figure 235: ShowStmt

OptFull:

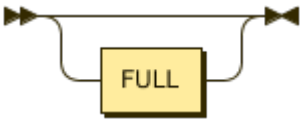

Figure 236: OptFull

4.1.5.77.2 Examples

```
mysql> SHOW PROCESSLIST;
+--
    ↪ ----+------+-----------+------+---------+------+-------+------------------+
    ↪
| Id   | User | Host      | db   | Command | Time | State | Info             |
+--
    ↪ ----+------+-----------+------+---------+------+-------+------------------+
    ↪
|    1 | root | 127.0.0.1 | test | Query | 0 | 2     | SHOW PROCESSLIST |
|    2 | root | 127.0.0.1 |      | Sleep |    4 | 2     |                  |
+--
    ↪ ----+------+-----------+------+---------+------+-------+------------------+
    ↪
2 rows in set (0.00 sec)
```

4.1.5.77.3 MySQL compatibility

- The `State` column in TiDB is non-descriptive. Representing state as a single value is more complex in TiDB, since queries are executed in parallel and each goroutine will have a different state at any one time.

4.1.5.77.4 See also

- KILL [TIDB]

4.1.5.78 SHOW SCHEMAS

This statement is an alias to `SHOW DATABASES`. It is included for compatibility with MySQL.

4.1.5.79 SHOW [GLOBAL|SESSION] STATUS

This statement is included for compatibility with MySQL. It has no effect on TiDB, which uses Prometheus and Grafana for centralized metrics collection instead of `SHOW STATUS`.

4.1.5.79.1 Synopsis

ShowStmt:

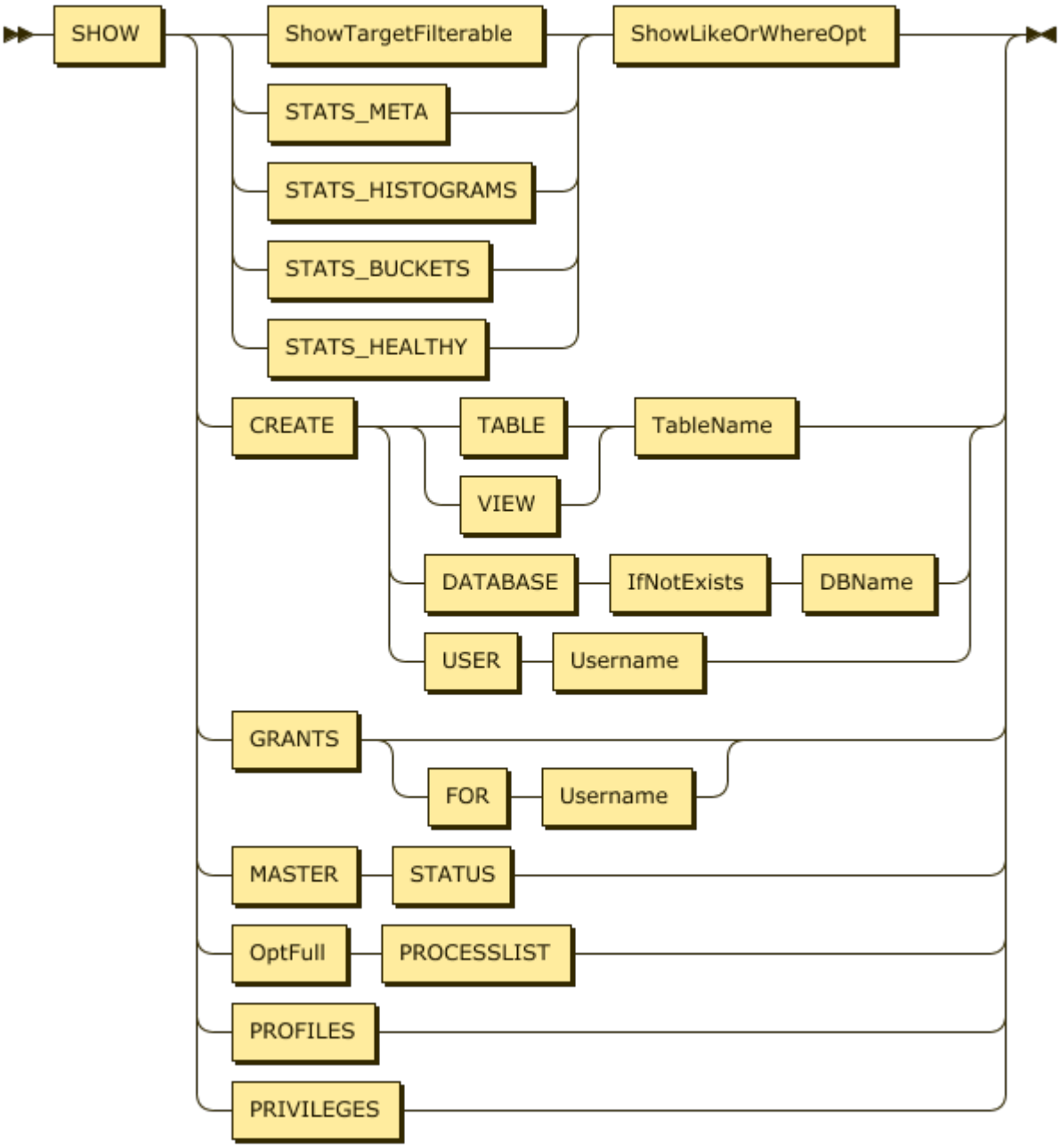

Figure 237: ShowStmt

ShowTargetFilterable:

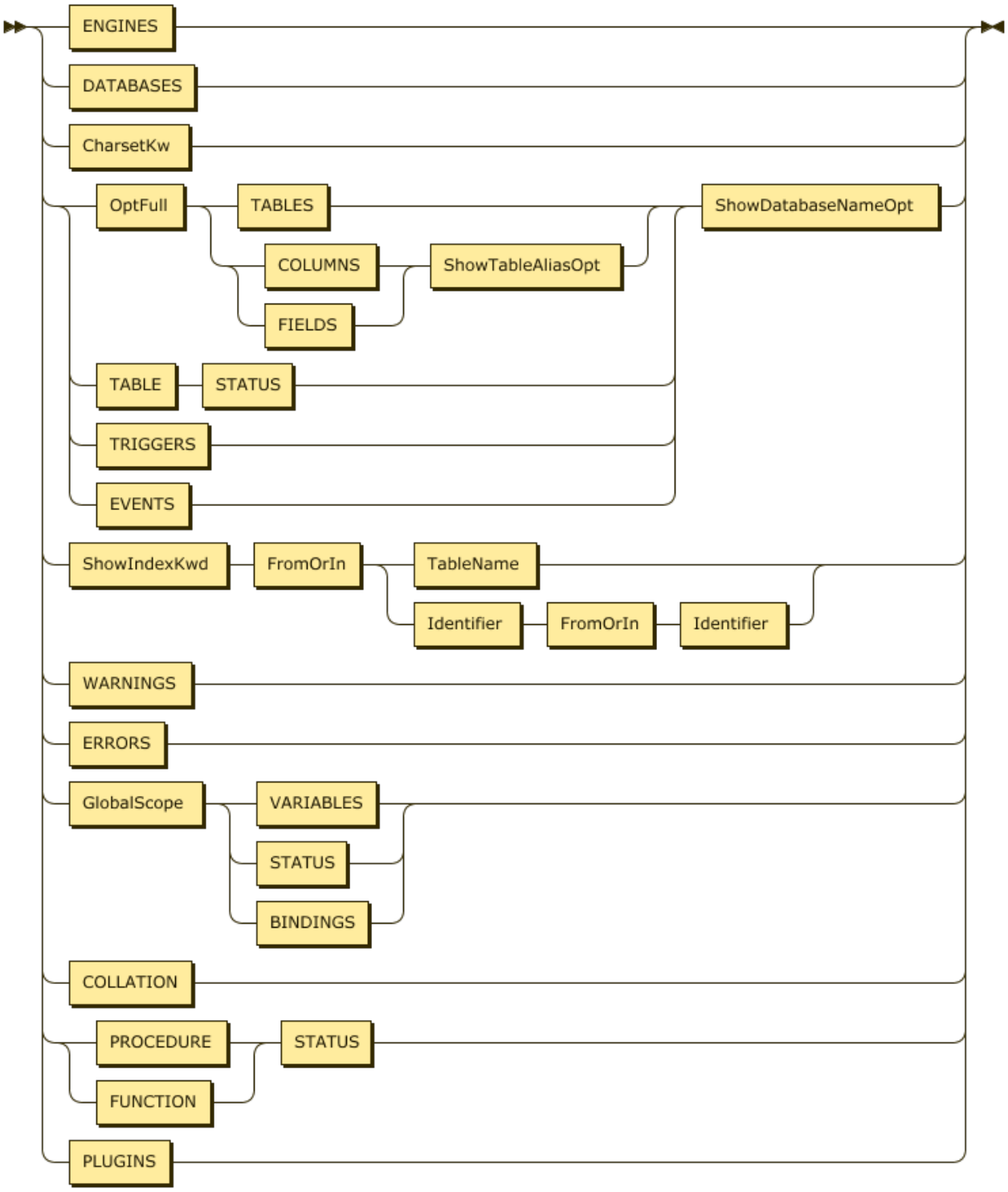

Figure 238: ShowTargetFilterable

GlobalScope:

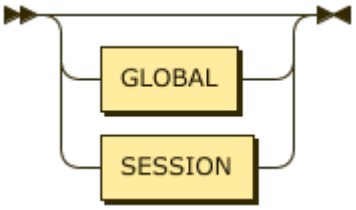

Figure 239: GlobalScope

4.1.5.79.2 Examples

```
mysql> show status;
+--------------------+--------------------------------------+
| Variable_name      | Value                                |
+--------------------+--------------------------------------+
| Ssl_cipher_list    |                                      |
| server_id          | 93e2e07d-6bb4-4a1b-90b7-e035fae154fe |
| ddl_schema_version | 141                                  |
| Ssl_verify_mode    | 0                                    |
| Ssl_version        |                                      |
| Ssl_cipher         |                                      |
+--------------------+--------------------------------------+
6 rows in set (0.01 sec)

mysql> show global status;
+--------------------+--------------------------------------+
| Variable_name      | Value                                |
+--------------------+--------------------------------------+
| Ssl_cipher         |                                      |
| Ssl_cipher_list    |                                      |
| Ssl_verify_mode    | 0                                    |
| Ssl_version        |                                      |
| server_id          | 93e2e07d-6bb4-4a1b-90b7-e035fae154fe |
| ddl_schema_version | 141                                  |
+--------------------+--------------------------------------+
6 rows in set (0.00 sec)
```

4.1.5.79.3 MySQL compatibility

- This statement is included only for compatibility with MySQL.

4.1.5.79.4 See also

- FLUSH STATUS

4.1.5.80 SHOW [FULL] TABLES

This statement shows a list of tables and views in the currently selected database. The optional keyword `FULL` indicates if a table is of type `BASE TABLE` or `VIEW`.

To show tables in a different database, use `SHOW TABLES IN DatabaseName`.

4.1.5.80.1 Synopsis

ShowStmt:

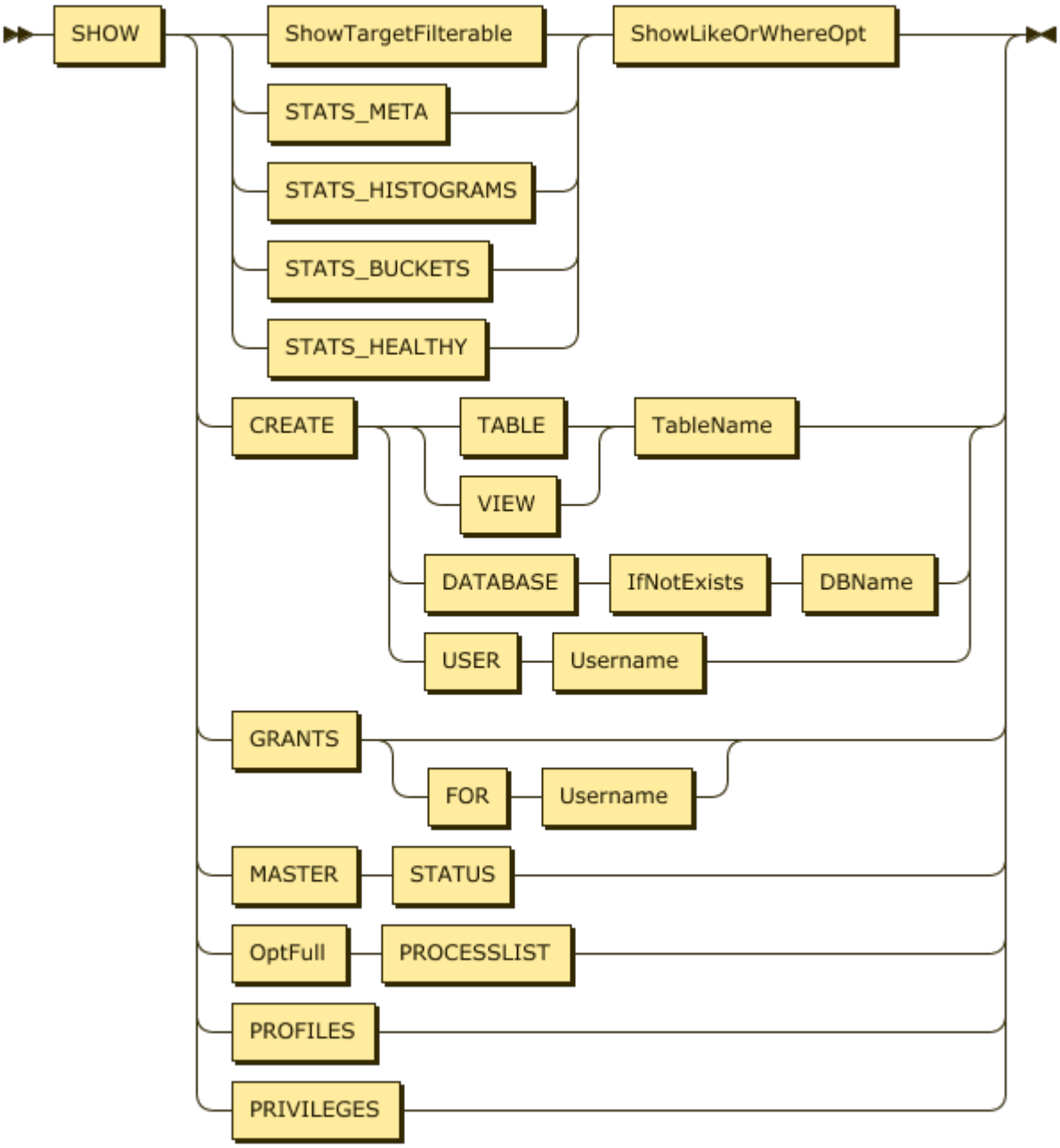

Figure 240: ShowStmt

ShowTargetFilterable:

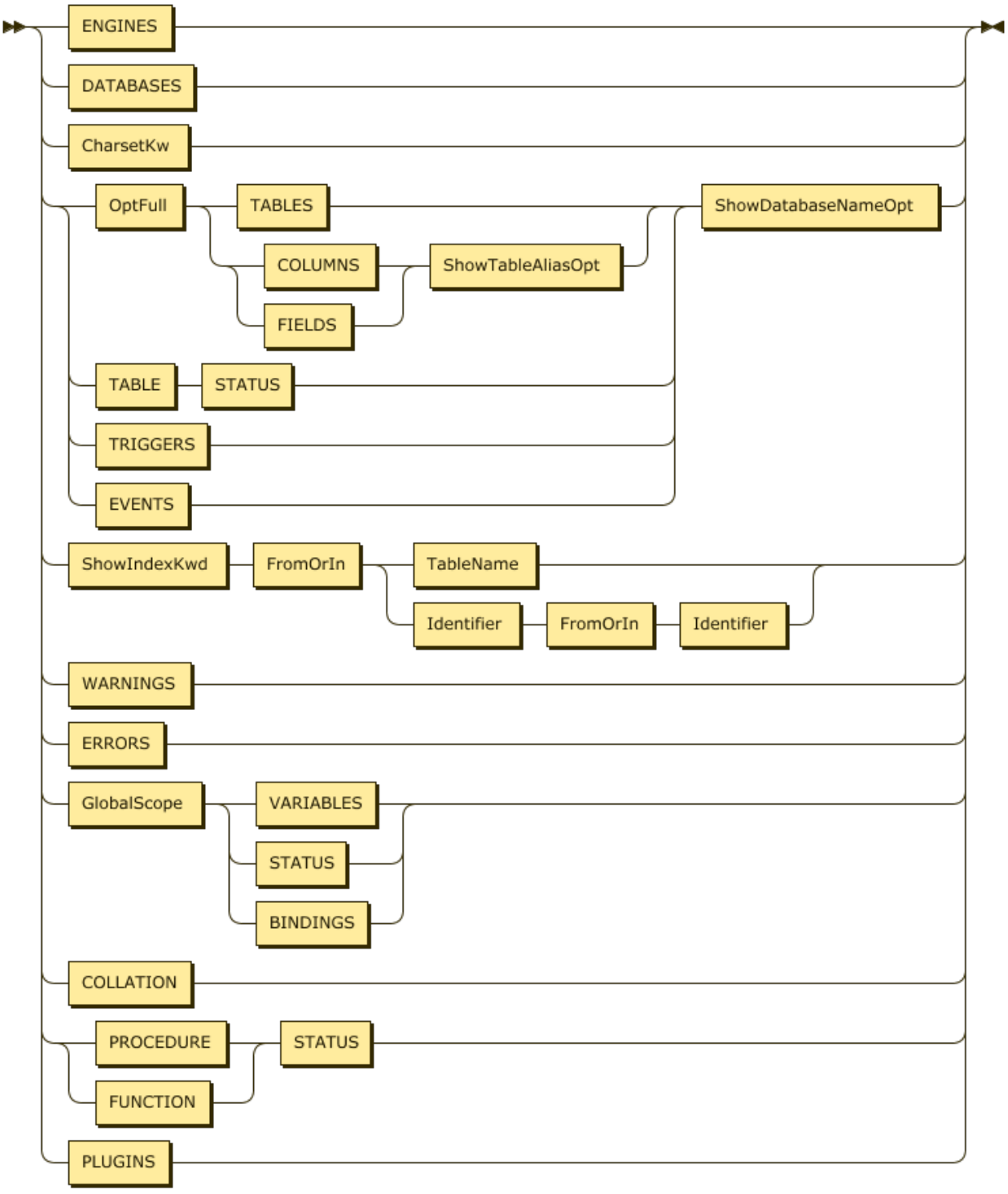

Figure 241: ShowTargetFilterable

ShowDatabaseNameOpt:

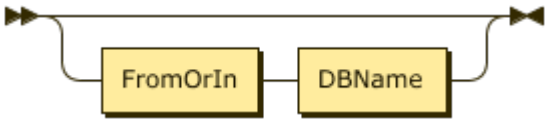

Figure 242: ShowDatabaseNameOpt

4.1.5.80.2 Examples

```
mysql> CREATE TABLE t1 (a int);
Query OK, 0 rows affected (0.12 sec)

mysql> CREATE VIEW v1 AS SELECT 1;
Query OK, 0 rows affected (0.10 sec)

mysql> SHOW TABLES;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
| v1             |
+----------------+
2 rows in set (0.00 sec)

mysql> SHOW FULL TABLES;
+----------------+------------+
| Tables_in_test | Table_type |
+----------------+------------+
| t1             | BASE TABLE |
| v1             | VIEW       |
+----------------+------------+
2 rows in set (0.00 sec)

mysql> SHOW TABLES IN mysql;
+----------------------+
| Tables_in_mysql      |
+----------------------+
| GLOBAL_VARIABLES     |
| bind_info            |
| columns_priv         |
| db                   |
| default_roles        |
| gc_delete_range      |
| gc_delete_range_done |
```

```
| help_topic         |
| role_edges         |
| stats_buckets      |
| stats_feedback     |
| stats_histograms   |
| stats_meta         |
| tables_priv        |
| tidb               |
| user               |
+--------------------+
16 rows in set (0.00 sec)
```

4.1.5.80.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.80.4 See also

- CREATE TABLE
- DROP TABLE
- SHOW CREATE TABLE

4.1.5.81 SHOW TABLE REGIONS

The `SHOW TABLE REGIONS` statement is used to show the Region information of a table in TiDB.

4.1.5.81.1 Synopsis

```
SHOW TABLE [table_name] REGIONS [WhereClauseOptional];
SHOW TABLE [table_name] INDEX [index_name] REGIONS [WhereClauseOptional];
```

Executing `SHOW TABLE REGIONS` returns the following columns:

- `REGION_ID`: The Region ID.
- `START_KEY`: The start key of the Region.
- `END_KEY`: The end key of the Region.
- `LEADER_ID`: The Leader ID of the Region.
- `LEADER_STORE_ID`: The ID of the store (TiKV) where the Region leader is located.
- `PEERS`: The IDs of all Region replicas.
- `SCATTERING`: Whether the Region is being scheduled. `1` means true.
- `WRITTEN_BYTES`: The estimated amount of data written into the Region within one heartbeat cycle. The unit is byte.

- READ_BYTES: The estimated amount of data read from the Region within one heartbeat cycle. The unit is byte.
- APPROXIMATE_SIZE(MB): The estimated amount of data in the Region. The unit is megabytes (MB).
- APPROXIMATE_KEYS: The estimated number of Keys in the Region.

> **Note:**
>
> The values of WRITTEN_BYTES, READ_BYTES, APPROXIMATE_SIZE(MB), APPROXIMATE_KEYS are not accurate data. They are estimated data from PD based on the heartbeat information that PD receives from the Region.

4.1.5.81.2 Examples

Create an example table with enough data that fills a few Regions:

```
CREATE TABLE t1 (
 id INT NOT NULL PRIMARY KEY auto_increment,
 b INT NOT NULL,
 pad1 VARBINARY(1024),
 pad2 VARBINARY(1024),
 pad3 VARBINARY(1024)
);
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM dual;
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM t1 a JOIN t1 b JOIN t1 c
    ↪ LIMIT 10000;
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM t1 a JOIN t1 b JOIN t1 c
    ↪ LIMIT 10000;
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM t1 a JOIN t1 b JOIN t1 c
    ↪ LIMIT 10000;
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM t1 a JOIN t1 b JOIN t1 c
    ↪ LIMIT 10000;
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM t1 a JOIN t1 b JOIN t1 c
    ↪ LIMIT 10000;
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM t1 a JOIN t1 b JOIN t1 c
    ↪ LIMIT 10000;
```

```
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM t1 a JOIN t1 b JOIN t1 c
    ↪ LIMIT 10000;
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM t1 a JOIN t1 b JOIN t1 c
    ↪ LIMIT 10000;
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM t1 a JOIN t1 b JOIN t1 c
    ↪ LIMIT 10000;
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM t1 a JOIN t1 b JOIN t1 c
    ↪ LIMIT 10000;
INSERT INTO t1 SELECT NULL, FLOOR(RAND()*1000), RANDOM_BYTES(1024),
    ↪ RANDOM_BYTES(1024), RANDOM_BYTES(1024) FROM t1 a JOIN t1 b JOIN t1 c
    ↪ LIMIT 10000;
SELECT SLEEP(5);
SHOW TABLE t1 REGIONS;
```

The output should show that the table is split into Regions. The REGION_ID, START_KEY and END_KEY may not match exactly:

```
...
mysql> SHOW TABLE t1 REGIONS;
+--
    ↪ ---------+---------------+---------------+-----------+------------------+-------+---
    ↪
| REGION_ID | START_KEY | END_KEY     | LEADER_ID | LEADER_STORE_ID | PEERS |
    ↪  SCATTERING | WRITTEN_BYTES | READ_BYTES | APPROXIMATE_SIZE(MB) |
    ↪ APPROXIMATE_KEYS |
+--
    ↪ ---------+---------------+---------------+-----------+------------------+-------+---
    ↪
|        94 | t_75_       | t_75_r_31717 |      95 |               1 | 95    |
    ↪          0 |             0 |          0 |                  112 |
    ↪ 207465 |
|        96 | t_75_r_31717 | t_75_r_63434 |    97 |               1 | 97    |
    ↪          0 |             0 |          0 |                   97 |
    ↪ 0 |
|         2 | t_75_r_63434 |            |         3 |               1 | 3     |
    ↪          0 |     269323514 | 66346110 |                  245 |
    ↪ 162020 |
+--
    ↪ ---------+---------------+---------------+-----------+------------------+-------+---
    ↪
3 rows in set (0.00 sec)
```

In the output above, a `START_KEY` of `t_75_r_31717` and `END_KEY` of `t_75_r_63434` shows that data with a PRIMARY KEY between `31717` and `63434` is stored in this Region. The prefix `t_75_` indicates that this is the Region for a table (`t`) which has an internal table ID of `75`. An empty key value for `START_KEY` or `END_KEY` indicates negative infinity or positive infinity respectively.

TiDB automatically rebalances Regions as needed. For manual rebalancing, use the `SPLIT TABLE REGION` statement:

```
mysql> SPLIT TABLE t1 BETWEEN (31717) AND (63434) REGIONS 2;
+--------------------+----------------------+
| TOTAL_SPLIT_REGION | SCATTER_FINISH_RATIO |
+--------------------+----------------------+
|                  1 |                    1 |
+--------------------+----------------------+
1 row in set (42.34 sec)

mysql> SHOW TABLE t1 REGIONS;
+--
   ↪ ---------+-------------+-------------+----------+----------------+-------+--
   ↪
| REGION_ID | START_KEY | END_KEY    | LEADER_ID | LEADER_STORE_ID | PEERS |
   ↪  SCATTERING | WRITTEN_BYTES | READ_BYTES | APPROXIMATE_SIZE(MB) |
   ↪ APPROXIMATE_KEYS |
+--
   ↪ ---------+-------------+-------------+----------+----------------+-------+--
   ↪
|       94 | t_75_      | t_75_r_31717 |     95 |             1 | 95   |
   ↪         0 |           0 |         0 |                112 |
   ↪ 207465 |
|       98 | t_75_r_31717 | t_75_r_47575 |   99 |             1 | 99   |
   ↪         0 |         1325 |         0 |                 53 |
   ↪ 12052 |
|       96 | t_75_r_47575 | t_75_r_63434 |   97 |             1 | 97   |
   ↪         0 |         1526 |         0 |                 48 |
   ↪ 0 |
|        2 | t_75_r_63434 |          |        3 |             1 | 3    |
   ↪         0 |           0 |   55752049 |                60 |
   ↪ 0 |
+--
   ↪ ---------+-------------+-------------+----------+----------------+-------+--
   ↪
4 rows in set (0.00 sec)
```

The above output shows that Region 96 was split, with a new Region 98 being created. The remaining Regions in the table were unaffected by the split operation. This is confirmed

by the output statistics:

- `TOTAL_SPLIT_REGION` indicates the number of newly split Regions. In this example, the number is 1.
- `SCATTER_FINISH_RATIO` indicates the rate at which the newly split Regions are successfully scattered. `1.0` means that all Regions are scattered.

For a more detailed example:

```
mysql> show table t regions;
+--
   ↪ ---------+-------------+-------------+----------+----------------+----------
   ↪
| REGION_ID | START_KEY | END_KEY    | LEADER_ID | LEADER_STORE_ID | PEERS
   ↪    | SCATTERING | WRITTEN_BYTES | READ_BYTES | APPROXIMATE_SIZE(MB) |
   ↪ APPROXIMATE_KEYS |
+--
   ↪ ---------+-------------+-------------+----------+----------------+----------
   ↪
| 102      | t_43_r      | t_43_r_20000 | 118    | 7              | 105, 118,
   ↪ 119 | 0       | 0           | 0         | 1                | 0
   ↪               |
| 106      | t_43_r_20000 | t_43_r_40000 | 120  | 7              | 107, 108,
   ↪ 120 | 0       | 23          | 0         | 1                | 0
   ↪               |
| 110      | t_43_r_40000 | t_43_r_60000 | 112  | 9              | 112, 113,
   ↪ 121 | 0       | 0           | 0         | 1                | 0
   ↪               |
| 114      | t_43_r_60000 | t_43_r_80000 | 122  | 7              | 115, 122,
   ↪ 123 | 0       | 35          | 0         | 1                | 0
   ↪               |
| 3        | t_43_r_80000 |           | 93       | 8              | 5, 73, 93
   ↪   | 0         | 0           | 0         | 1                | 0
   ↪               |
| 98       | t_43_       | t_43_r      | 99       | 1              | 99, 100,
   ↪ 101 | 0        | 0           | 0         | 1                 | 0
   ↪               |
+--
   ↪ ---------+-------------+-------------+----------+----------------+----------
   ↪
6 rows in set
```

In the above example:

- Table t corresponds to six Regions. In these Regions, `102`, `106`, `110`, `114`, and `3` store the row data and `98` stores the index data.

- For START_KEY and END_KEY of Region 102, t_43 indicates the table prefix and ID. _r is the prefix of the record data in table t. _i is the prefix of the index data.
- In Region 102, START_KEY and END_KEY mean that record data in the range of [-inf, 20000) is stored. In similar way, the ranges of data storage in Regions (103, 109, 113, 2) can also be calculated.
- Region 98 stores the index data. The start key of table t's index data is t_43_i, which is in the range of Region 98.

To check the Region that corresponds to table t in store 1, use the WHERE clause:

```
test> show table t regions where leader_store_id =1;
+--
   ↪ ---------+-----------+---------+-----------+----------------+--------------+---
   ↪
| REGION_ID | START_KEY | END_KEY | LEADER_ID | LEADER_STORE_ID | PEERS |
   ↪ SCATTERING | WRITTEN_BYTES | READ_BYTES | APPROXIMATE_SIZE(MB) |
   ↪ APPROXIMATE_KEYS |
+--
   ↪ ---------+-----------+---------+-----------+----------------+--------------+---
   ↪
| 98       | t_43_    | t_43_r | 99       | 1             | 99, 100, 101 | 0
   ↪       | 0            | 0         | 1                | 0              |
+--
   ↪ ---------+-----------+---------+-----------+----------------+--------------+---
   ↪
```

Use SPLIT TABLE REGION to split the index data into Regions. In the following example, the index data name of table t is split into two Regions in the range of [a,z].

```
test> split table t index name between ("a") and ("z") regions 2;
+--------------------+----------------------+
| TOTAL_SPLIT_REGION | SCATTER_FINISH_RATIO |
+--------------------+----------------------+
| 2                  | 1.0                  |
+--------------------+----------------------+
1 row in set
```

Now table t corresponds to seven Regions. Five of them (102, 106, 110, 114, 3) store the record data of table t and another two (135, 98) store the index data name.

```
test> show table t regions;
+--
   ↪ ---------+----------------------------+----------------------------+----------
   ↪
| REGION_ID | START_KEY                | END_KEY                    | LEADER_ID |
   ↪  LEADER_STORE_ID | PEERS    | SCATTERING | WRITTEN_BYTES | READ_BYTES
   ↪ | APPROXIMATE_SIZE(MB) | APPROXIMATE_KEYS |
```

```
+--
    ↪ ---------+----------------------------+----------------------------+---------
    ↪
| 102      | t_43_r                     | t_43_r_20000               | 118      |
    ↪ 7              | 105, 118, 119 | 0       | 0           | 0         | 1
    ↪                  | 0              |
| 106      | t_43_r_20000               | t_43_r_40000               | 120      |
    ↪ 7              | 108, 120, 126 | 0       | 0           | 0         | 1
    ↪                  | 0              |
| 110      | t_43_r_40000               | t_43_r_60000               | 112      |
    ↪ 9              | 112, 113, 121 | 0       | 0           | 0         | 1
    ↪                  | 0              |
| 114      | t_43_r_60000               | t_43_r_80000               | 122      |
    ↪ 7              | 115, 122, 123 | 0       | 35          | 0         | 1
    ↪                  | 0              |
| 3        | t_43_r_80000               |                            | 93       |
    ↪ 8              | 73, 93, 128 | 0         | 0           | 0         | 1
    ↪                  | 0              |
| 135      | t_43_i_1_                  | t_43_i_1_016d80000000000000 | 139  |
    ↪ 2              | 138, 139, 140 | 0       | 35          | 0         | 1
    ↪                  | 0              |
| 98       | t_43_i_1_016d80000000000000 | t_43_r                     | 99       |
    ↪ 1              | 99, 100, 101 | 0        | 0           | 0         | 1
    ↪                  | 0              |
+--
    ↪ ---------+----------------------------+----------------------------+---------
    ↪
7 rows in set
```

4.1.5.81.3 MySQL compatibility

This statement is a TiDB extension to MySQL syntax.

4.1.5.81.4 See also

- SPLIT REGION
- CREATE TABLE

4.1.5.82 SHOW TABLE STATUS

This statement shows various statistics about tables in TiDB. If the statistics appear out of date, it is recommended to run `ANALYZE TABLE`.

4.1.5.82.1 Synopsis

ShowStmt:

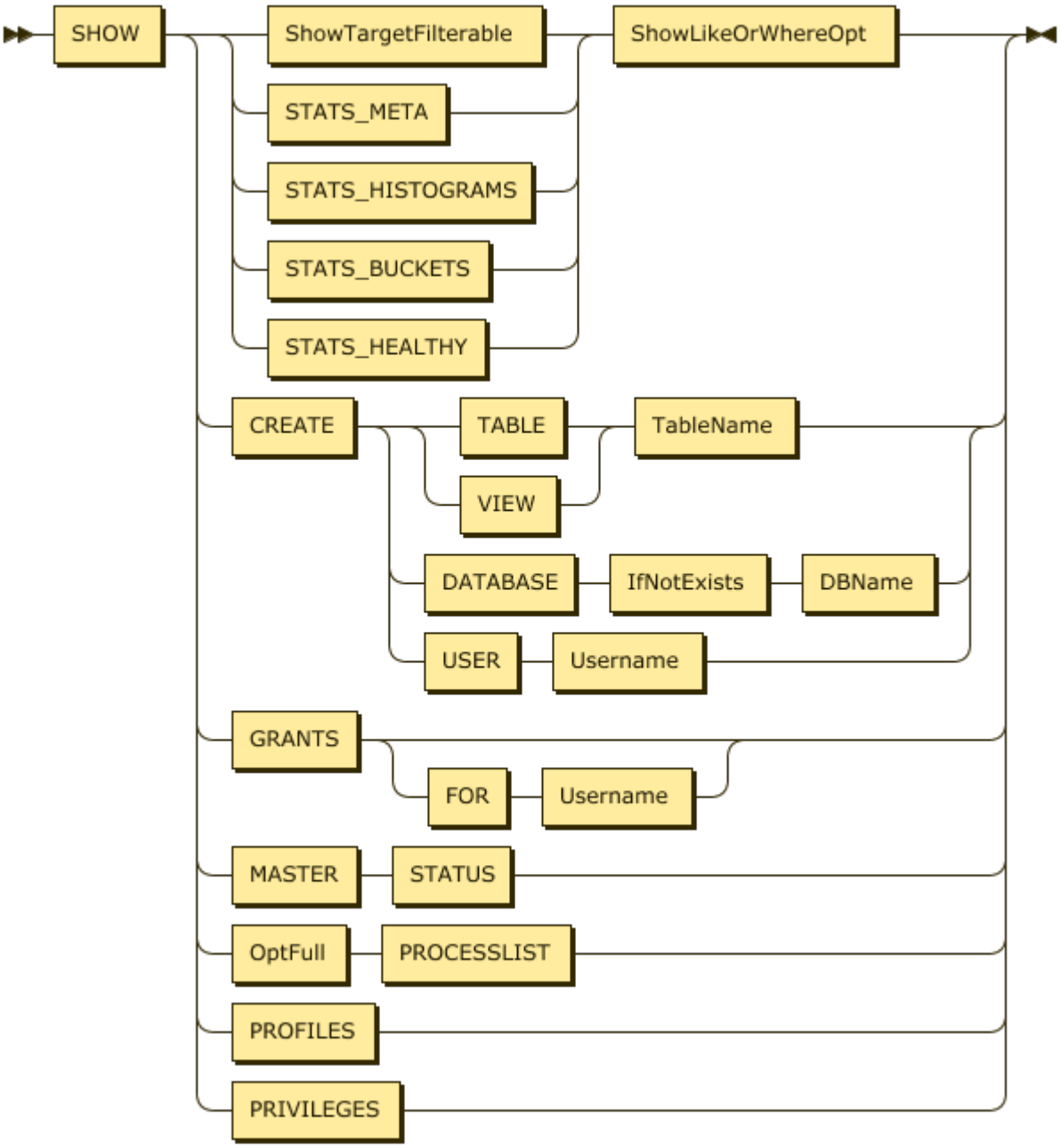

Figure 243: ShowStmt

ShowTargetFilterable:

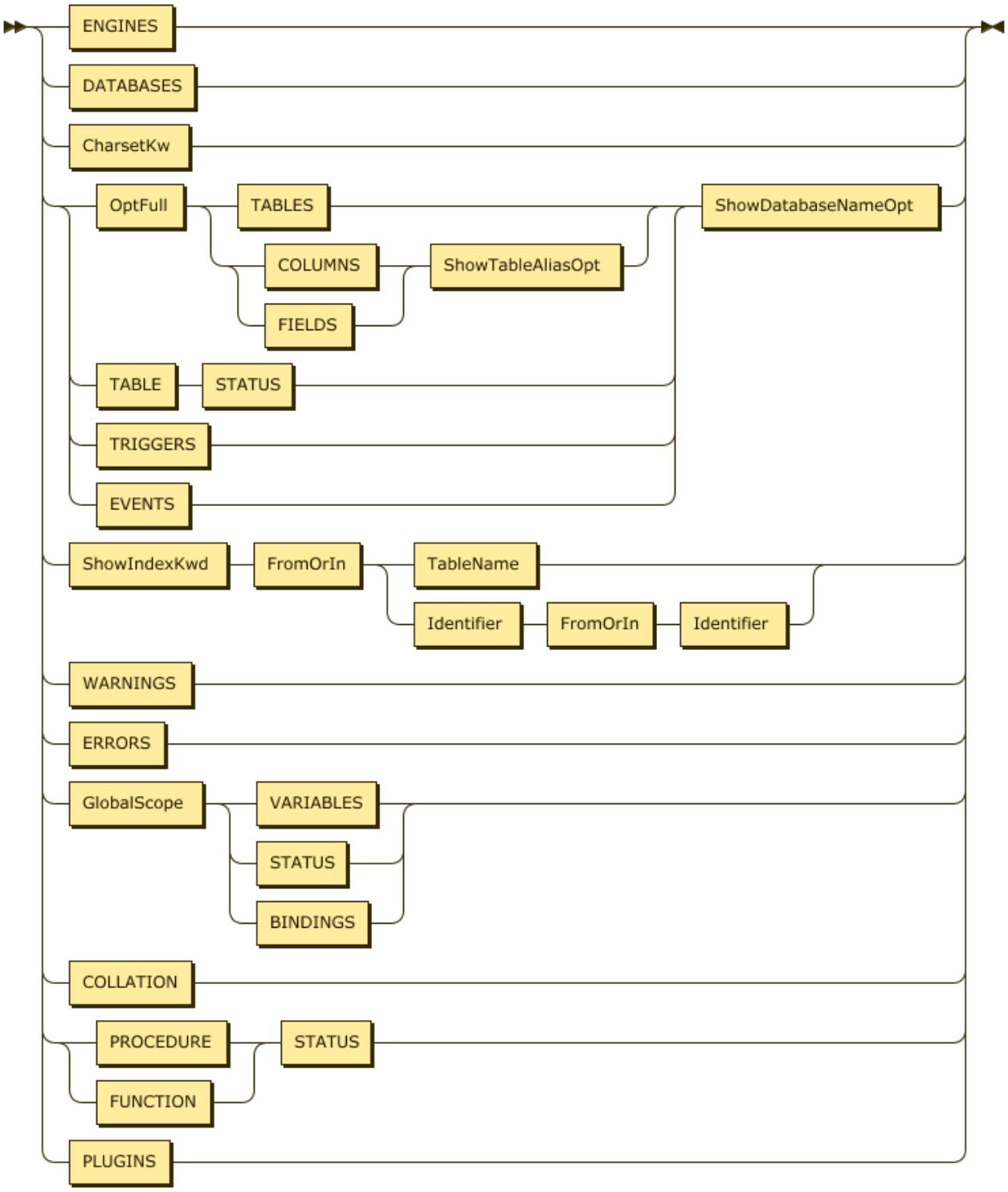

Figure 244: ShowTargetFilterable

ShowDatabaseNameOpt:

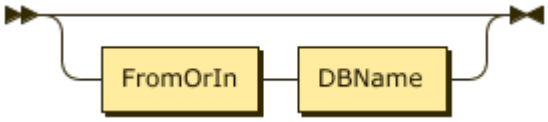

Figure 245: ShowDatabaseNameOpt

4.1.5.82.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 (c1) VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> SHOW TABLE STATUS LIKE 't1'\G
*************************** 1. row ***************************
           Name: t1
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 0
 Avg_row_length: 0
    Data_length: 0
Max_data_length: 0
   Index_length: 0
      Data_free: 0
 Auto_increment: 30001
    Create_time: 2019-04-19 08:32:06
    Update_time: NULL
     Check_time: NULL
      Collation: utf8mb4_bin
       Checksum:
 Create_options:
        Comment:
1 row in set (0.00 sec)

mysql> analyze table t1;
Query OK, 0 rows affected (0.12 sec)

mysql> SHOW TABLE STATUS LIKE 't1'\G
*************************** 1. row ***************************
           Name: t1
         Engine: InnoDB
```

```
       Version: 10
    Row_format: Compact
          Rows: 5
 Avg_row_length: 16
   Data_length: 80
Max_data_length: 0
   Index_length: 0
      Data_free: 0
 Auto_increment: 30001
   Create_time: 2019-04-19 08:32:06
   Update_time: NULL
    Check_time: NULL
     Collation: utf8mb4_bin
      Checksum:
 Create_options:
       Comment:
1 row in set (0.00 sec)
```

4.1.5.82.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.82.4 See also

- SHOW TABLES
- CREATE TABLE
- DROP TABLE
- SHOW CREATE TABLE

4.1.5.83 SHOW [GLOBAL|SESSION] VARIABLES

This statement shows a list of variables for the scope of either `GLOBAL` or `SESSION`. If no scope is specified, the default scope of `SESSION` will apply.

4.1.5.83.1 Synopsis

ShowStmt:

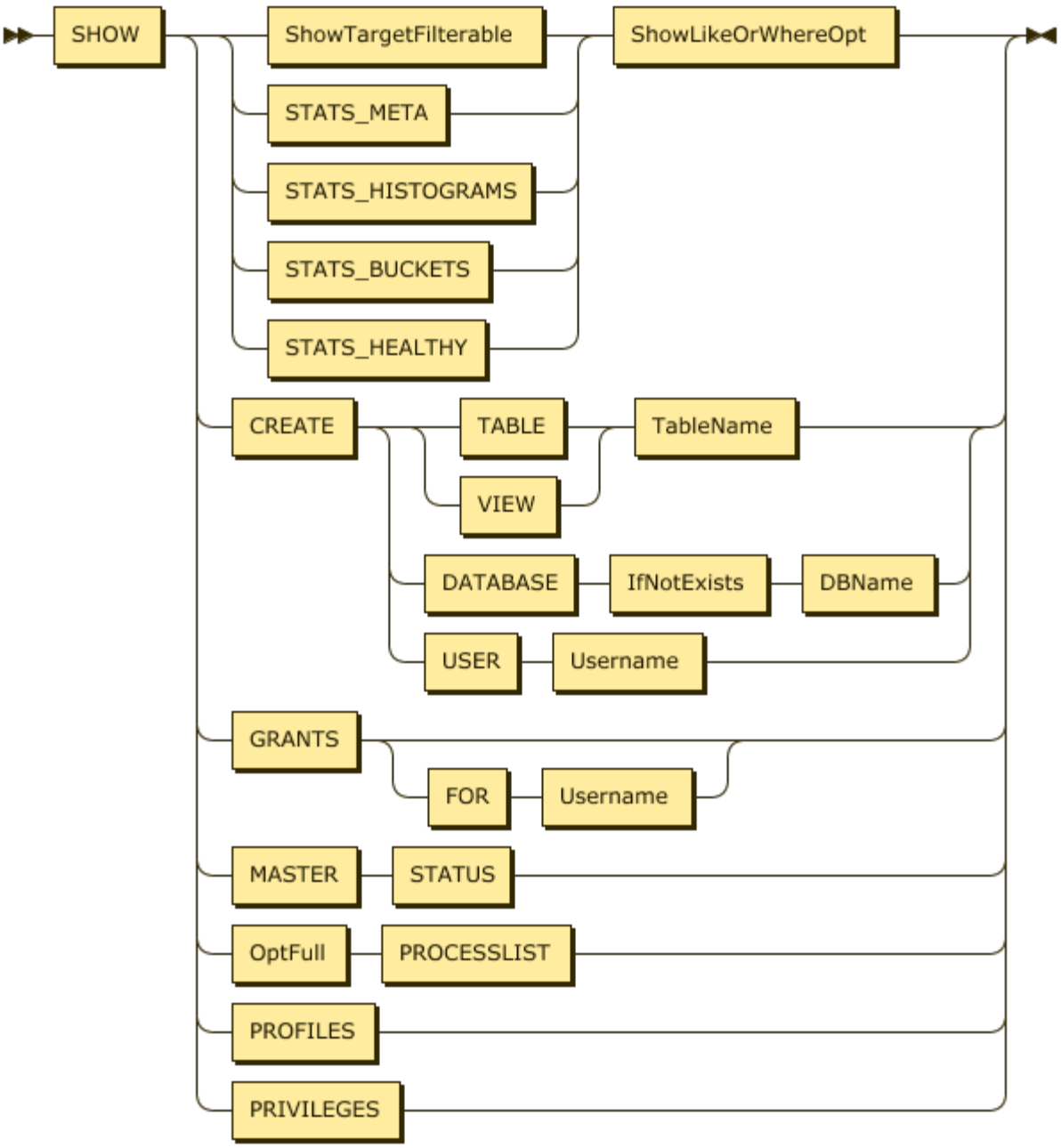

Figure 246: ShowStmt

ShowTargetFilterable:

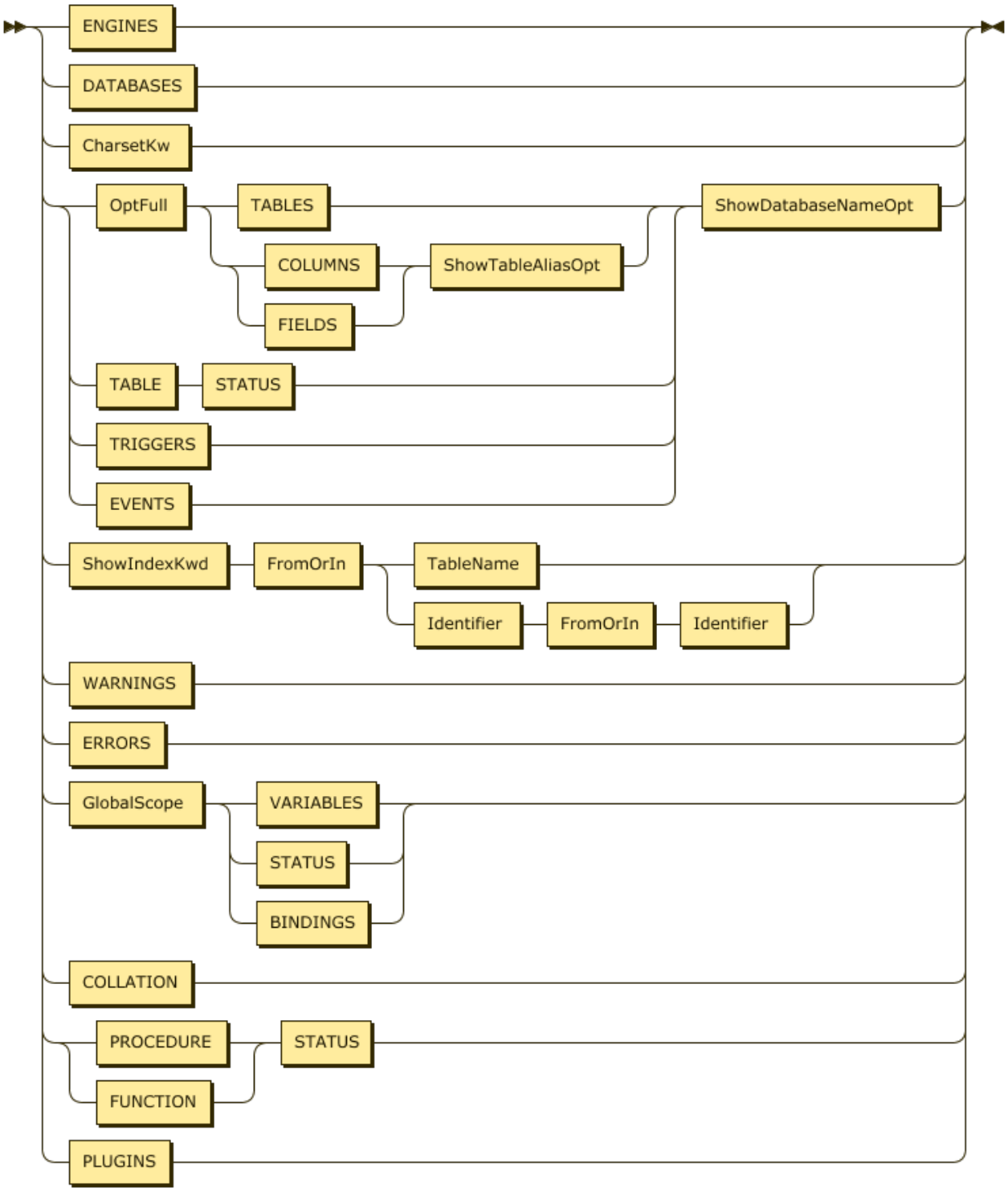

Figure 247: ShowTargetFilterable

GlobalScope:

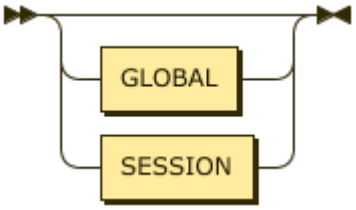

Figure 248: GlobalScope

4.1.5.83.2 Examples

```
mysql> SHOW GLOBAL VARIABLES LIKE 'tidb%';
+-----------------------------------+---------------+
| Variable_name                     | Value         |
+-----------------------------------+---------------+
| tidb_retry_limit                  | 10            |
| tidb_hashagg_final_concurrency    | 4             |
| tidb_disable_txn_auto_retry       | 1             |
| tidb_mem_quota_query              | 34359738368   |
| tidb_skip_isolation_level_check   | 0             |
| tidb_ddl_reorg_batch_size         | 1024          |
| tidb_constraint_check_in_place    | 0             |
| tidb_current_ts                   | 0             |
| tidb_opt_insubq_to_join_and_agg   | 1             |
| tidb_enable_window_function       | 1             |
| tidb_ddl_error_count_limit        | 512           |
| tidb_checksum_table_concurrency   | 4             |
| tidb_config                       |               |
| tidb_batch_insert                 | 0             |
| tidb_opt_join_reorder_threshold   | 0             |
| tidb_auto_analyze_end_time        | 23:59 +0000   |
| tidb_index_lookup_size            | 20000         |
| tidb_projection_concurrency       | 4             |
| tidb_opt_correlation_threshold    | 0.9           |
| tidb_enable_table_partition       | auto          |
| tidb_wait_split_region_timeout    | 300           |
| tidb_skip_utf8_check              | 0             |
| tidb_dml_batch_size               | 20000         |
| tidb_backoff_weight               | 2             |
| tidb_txn_mode                     |               |
| tidb_mem_quota_indexlookupjoin    | 34359738368   |
| tidb_hashagg_partial_concurrency  | 4             |
| tidb_enable_radix_join            | 0             |
| tidb_mem_quota_topn               | 34359738368   |
| tidb_hash_join_concurrency        | 5             |
| tidb_max_chunk_size               | 1024          |
```

```
| tidb_mem_quota_indexlookupreader | 34359738368 |
| tidb_expensive_query_time_threshold | 60     |
| tidb_enable_cascades_planner    | 0           |
| tidb_mem_quota_mergejoin        | 34359738368 |
| tidb_auto_analyze_ratio         | 0.5         |
| tidb_index_lookup_concurrency   | 4           |
| tidb_force_priority             | NO_PRIORITY |
| tidb_ddl_reorg_priority         | PRIORITY_LOW |
| tidb_index_lookup_join_concurrency | 4        |
| tidb_index_join_batch_size      | 25000       |
| tidb_index_serial_scan_concurrency | 1        |
| tidb_mem_quota_nestedloopapply  | 34359738368 |
| tidb_auto_analyze_start_time    | 00:00 +0000 |
| tidb_snapshot                   |             |
| tidb_low_resolution_tso         | 0           |
| tidb_batch_commit               | 0           |
| tidb_init_chunk_size            | 32          |
| tidb_build_stats_concurrency    | 4           |
| tidb_backoff_lock_fast          | 100         |
| tidb_optimizer_selectivity_level | 0          |
| tidb_batch_delete               | 0           |
| tidb_distsql_scan_concurrency   | 15          |
| tidb_scatter_region             | 0           |
| tidb_opt_correlation_exp_factor | 1           |
| tidb_ddl_reorg_worker_cnt       | 16          |
| tidb_wait_split_region_finish   | 1           |
| tidb_mem_quota_sort             | 34359738368 |
| tidb_general_log                | 0           |
| tidb_opt_write_row_id           | 0           |
| tidb_check_mb4_value_in_utf8    | 1           |
| tidb_enable_fast_analyze        | 0           |
| tidb_slow_query_file            | tidb-slow.log |
| tidb_slow_log_threshold         | 300         |
| tidb_opt_agg_push_down          | 0           |
| tidb_mem_quota_hashjoin         | 34359738368 |
| tidb_query_log_max_len          | 2048        |
+-------------------------------------+-------------+
68 rows in set (0.01 sec)

mysql> SHOW GLOBAL VARIABLES LIKE 'time_zone%';
+---------------+--------+
| Variable_name | Value |
+---------------+--------+
| time_zone    | SYSTEM |
+---------------+--------+
```

```
1 row in set (0.00 sec)
```

4.1.5.83.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.83.4 See also

- SET [GLOBAL|SESSION]

4.1.5.84 SHOW WARNINGS

This statement shows a list of warnings that occurred for previously executed statements in the current client connection. As in MySQL, the `sql_mode` impacts which statements will cause errors vs. warnings considerably.

4.1.5.84.1 Synopsis

ShowStmt:

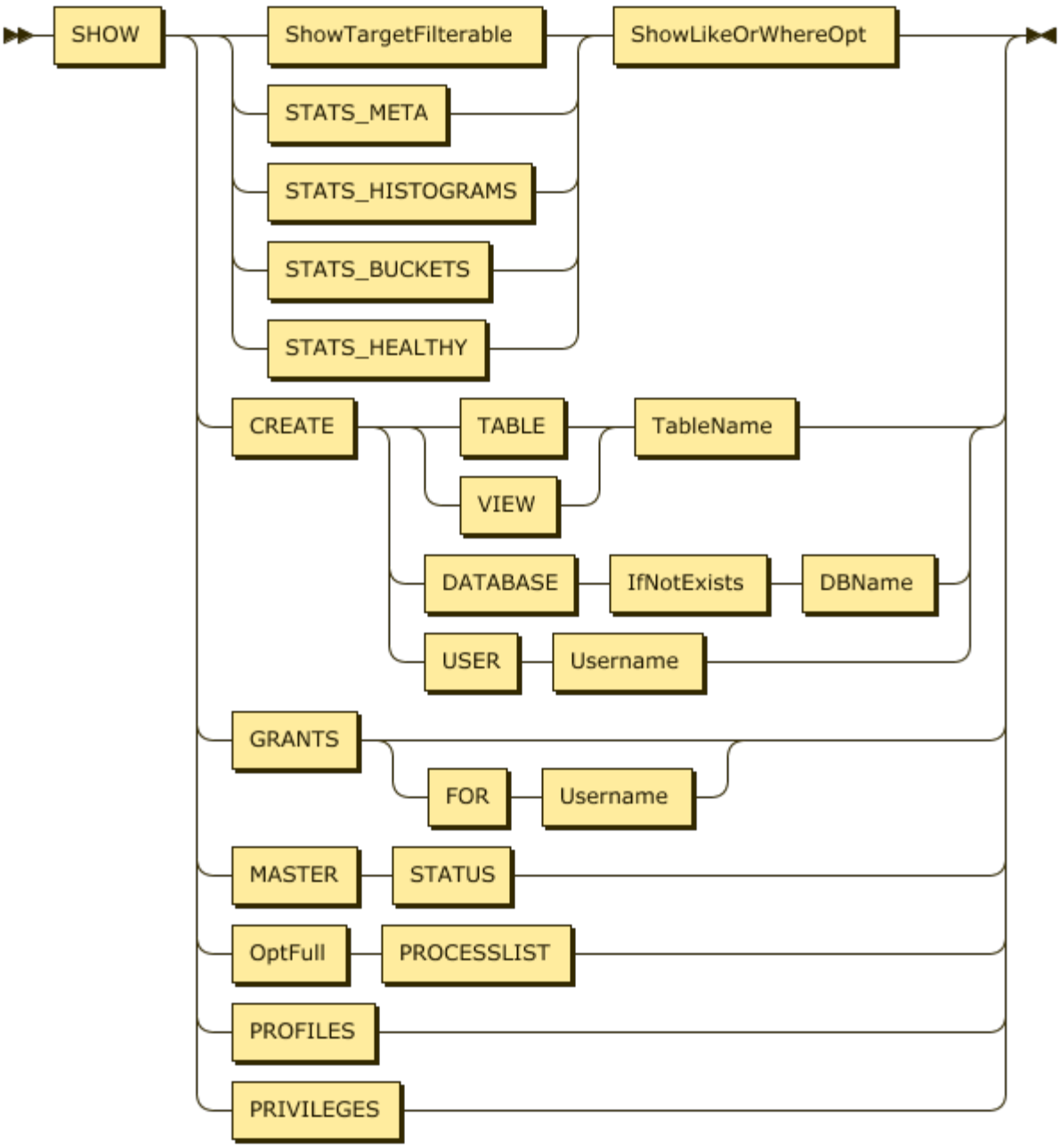

Figure 249: ShowStmt

ShowTargetFilterable:

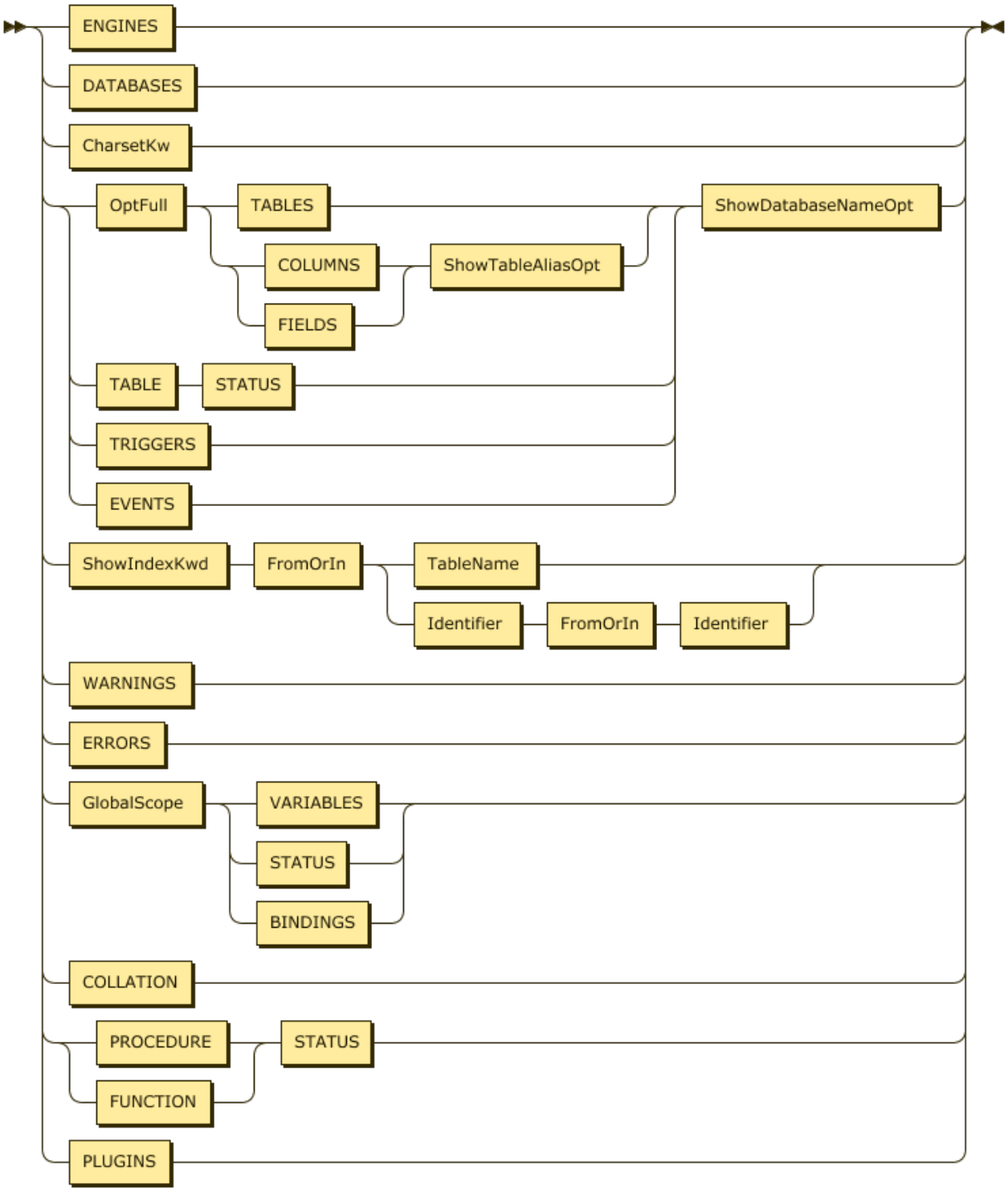

Figure 250: ShowTargetFilterable

4.1.5.84.2 Examples

```
mysql> CREATE TABLE t1 (a INT UNSIGNED);
```

```
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 VALUES (0);
Query OK, 1 row affected (0.02 sec)

mysql> SELECT 1/a FROM t1;
+------+
| 1/a  |
+------+
| NULL |
+------+
1 row in set, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;
+---------+------+---------------+
| Level   | Code | Message       |
+---------+------+---------------+
| Warning | 1365 | Division by 0 |
+---------+------+---------------+
1 row in set (0.00 sec)

mysql> INSERT INTO t1 VALUES (-1);
ERROR 1264 (22003): Out of range value for column 'a' at row 1
mysql> SELECT * FROM t1;
+------+
| a    |
+------+
|    0 |
+------+
1 row in set (0.00 sec)

mysql> SET sql_mode='';
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t1 VALUES (-1);
Query OK, 1 row affected, 1 warning (0.01 sec)

mysql> SHOW WARNINGS;
+---------+------+---------------------------+
| Level   | Code | Message                   |
+---------+------+---------------------------+
| Warning | 1690 | constant -1 overflows int |
+---------+------+---------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM t1;
+------+
| a    |
+------+
|    0 |
|    0 |
+------+
2 rows in set (0.00 sec)
```

4.1.5.84.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.84.4 See also

- SHOW ERRORS

4.1.5.85 Split Region

For each new table created in TiDB, one Region is segmented by default to store the data of this table. This default behavior is controlled by `split-table` in the configuration file. When the data in this Region exceeds the default Region size limit, the Region starts to split into two.

In the above case, because there is only one Region at the beginning, all write requests occur on the TiKV where the Region is located. If there are a large number of writes for the newly created table, hotspots are caused.

To solve the hotspot problem in the above scenario, TiDB introduces the pre-split function, which can pre-split multiple Regions for a certain table according to the specified parameters and scatter them to each TiKV node.

4.1.5.85.1 Usage of Split Region

There are two types of Split Region syntax:

```
SPLIT TABLE table_name [INDEX index_name] BETWEEN (lower_value) AND (
    ↪ upper_value) REGIONS region_num
```

`BETWEEN lower_value AND upper_value REGIONS region_num` defines the upper boundary, the lower boundary, and the Region amount. Then the current region will be evenly spilt into the number of regions (as specified in `region_num`) between the upper and lower boundaries.

```
SPLIT TABLE table_name [INDEX index_name] BY (value_list) [, (value_list)]
    ↪ ...
```

`BY value_list···` specifies a series of points manually, based on which the current Region is spilt. It is suitable for scenarios with unevenly distributed data.

The following example shows the result of the `SPLIT` statement:

```
+--------------------+---------------------+
| TOTAL_SPLIT_REGION | SCATTER_FINISH_RATIO |
+--------------------+---------------------+
| 4                  | 1.0                 |
+--------------------+---------------------+
```

- `TOTAL_SPLIT_REGION`: the number of newly split Regions.
- `SCATTER_FINISH_RATIO`: the completion rate of scattering for newly split Regions. `1.0` means that all Regions are scattered. `0.5` means that only half of the Regions are scattered and the rest are being scattered.

Note:

The following two session variables might affect the behavior of the `SPLIT` statement:

- `tidb_wait_split_region_finish`: It might take a while to scatter the Regions. This duration depends on PD scheduling and TiKV loads. This variable is used to control when executing the `SPLIT REGION` statement whether to return the results to the client until all Regions are scattered. If its value is set to `1` (by default), TiDB returns the results only after the scattering is completed. If its value is set to `0`, TiDB returns the results regardless of the scattering status.
- `tidb_wait_split_region_timeout`: This variable is to set the execution timeout of the `SPLIT REGION` statement, in seconds. The default value is 300s. If the `split` operation is not completed within the duration, TiDB returns a timeout error.

Split Table Region

The key of row data in each table is encoded by `table_id` and `row_id`. The format is as follows:

```
t[table_id]_r[row_id]
```

For example, when `table_id` is 22 and `row_id` is 11:

```
t22_r11
```

Row data in the same table have the same `table_id`, but each row has its unique `row_id` that can be used for Region split.

Even Split

Because `row_id` is an integer, the value of the key to be split can be calculated according to the specified `lower_value`, `upper_value`, and `region_num`. TiDB first calculates the step value (`step = (upper_value - lower_value)/num`). Then split will be done evenly per each "step" between `lower_value` and `upper_value` to generate the number of Regions as specified by `num`.

For example, if you want 16 evenly split Regions split from key range`minInt64~maxInt64` for table t, you can use this statement:

```
SPLIT TABLE t BETWEEN (-9223372036854775808) AND (9223372036854775807)
    ↪ REGIONS 16;
```

This statement splits table t into 16 Regions between minInt64 and maxInt64. If the given primary key range is smaller than the specified one, for example, 0~1000000000, you can use 0 and 1000000000 take place of minInt64 and maxInt64 respectively to split Regions.

```
SPLIT TABLE t BETWEEN (0) AND (1000000000) REGIONS 16;
```

Uneven split

If the known data is unevenly distributed, and you want a Region to be split respectively in key ranges -inf ~ 10000, 10000 ~ 90000, and 90000 ~ +inf, you can achieve this by setting fixed points, as shown below:

```
SPLIT TABLE t BY (10000), (90000);
```

Split index Region

The key of the index data in the table is encoded by `table_id`, `index_id`, and the value of the index column. The format is as follows:

```
t[table_id]_i[index_id][index_value]
```

For example, when `table_id` is 22, `index_id` is 5, and `index_value` is abc:

```
t22_i5abc
```

The `table_id` and `index_id` of the same index data in one table is the same. To split index Regions, you need to split Regions based on `index_value`.

Even Spilt

The way to split index evenly works the same as splitting data evenly. However, calculating the value of step is more complicated, because `index_value` might not be an integer.

The values of `upper` and `lower` are encoded into a byte array firstly. After removing the longest common prefix of `lower` and `upper` byte array, the first 8 bytes of `lower` and `upper`

are converted into the uint64 format. Then `step = (upper - lower)/num` is calculated. After that, the calculated step is encoded into a byte array, which is appended to the longest common prefix of the `lower` and `upper` byte array for index split. Here is an example:

If the column of the `idx` index is of the integer type, you can use the following SQL statement to split index data:

```
SPLIT TABLE t INDEX idx BETWEEN (-9223372036854775808) AND
    ↪ (9223372036854775807) REGIONS 16;
```

This statement splits the Region of index idx in table t into 16 Regions from `minInt64` to `maxInt64`.

If the column of index idx1 is of varchar type, and you want to split index data by prefix letters.

```
SPLIT TABLE t INDEX idx1 BETWEEN ("a") AND ("z") REGIONS 25;
```

This statement splits index idx1 into 25 Regions from a~z. The range of Region 1 is `[minIndexValue, b)`; the range of Region 2 is `[b, c)`; … the range of Region 25 is `[y,
↪ minIndexValue]`. For the `idx` index, data with the `a` prefix is written into Region 1, and data with the `b` prefix is written into Region 2, and so on.

In the split method above, both data with the `y` and `z` prefixes are written into Region 25, because the upper bound is not `z`, but `{` (the character next to `z` in ASCII). Therefore, a more accurate split method is as follows:

```
SPLIT TABLE t INDEX idx1 BETWEEN ("a") AND ("{") REGIONS 26;
```

This statement splits index idx1 of the table `t` into 26 Regions from a~`{`. The range of Region 1 is `[minIndexValue, b)`; the range of Region 2 is `[b, c)`; … the range of Region 25 is `[y, z)`, and the range of Region 26 is `[z, maxIndexValue)`.

If the column of index `idx2` is of time type like timestamp/datetime, and you want to split index Region by year:

```
SPLIT TABLE t INDEX idx2 BETWEEN ("2010-01-01 00:00:00") AND ("2020-01-01
    ↪ 00:00:00") REGIONS 10;
```

This statement splits the Region of index `idx2` in table `t` into 10 Regions from `2010-01-01 00:00:00` to `2020-01-01 00:00:00`. The range of Region 1 is `[minIndexValue
↪ , 2011-01-01 00:00:00)`; the range of Region 2 is `[2011-01-01 00:00:00,
↪ 2012-01-01 00:00:00)` and so on.

If you want to split the index Region by day, see the following example:

```
SPLIT TABLE t INDEX idx2 BETWEEN ("2020-06-01 00:00:00") AND ("2020-07-01
    ↪ 00:00:00") REGIONS 30;
```

This statement splits the data of June 2020 of index `idx2` in table `t` into 30 Regions, each Region representing 1 day.

Region split methods for other types of index columns are similar.

For data Region split of joint indexes, the only difference is that you can specify multiple columns values.

For example, index `idx3 (a, b)` contains 2 columns, with column `a` of timestamp type and column `b` int. If you just want to do a time range split according to column `a`, you can use the SQL statement for splitting time index of a single column. In this case, do not specify the value of column `b` in `lower_value` and `upper_velue`.

```
SPLIT TABLE t INDEX idx3 BETWEEN ("2010-01-01 00:00:00") AND ("2020-01-01
    ↪ 00:00:00") REGIONS 10;
```

Within the same range of time, if you want to do one more split according to column b column. Just specify the value for column b when splitting.

```
SPLIT TABLE t INDEX idx3 BETWEEN ("2010-01-01 00:00:00", "a") AND ("
    ↪ 2010-01-01 00:00:00", "z") REGIONS 10;
```

This statement splits 10 Regions in the range of a~z according to the value of column b, with the same time prefix as column a. If the value specified for column a is different, the value of column b might not be used in this case.

Uneven Split

Index data can also be split by specified index values.

For example, there is `idx4 (a,b)`, with column `a` of the varchar type and column `b` of the timestamp type.

```
SPLIT TABLE t1 INDEX idx4 BY ("a", "2000-01-01 00:00:01"), ("b", "
    ↪ 2019-04-17 14:26:19"), ("c", "");
```

This statement specifies 3 values to split 4 Regions. The range of each Region is as follows:

```
region1 [ minIndexValue           , ("a", "2000-01-01 00:00:01"))
region2 [("a", "2000-01-01 00:00:01") , ("b", "2019-04-17 14:26:19"))
region3 [("b", "2019-04-17 14:26:19") , ("c", "")           )
region4 [("c", "")                , maxIndexValue           )
```

4.1.5.85.2 pre_split_regions

To have evenly split Regions when a table is created, it is recommended you use `SHARD_ROW_ID_BITS` together with `PRE_SPLIT_REGIONS`. When a table is created successfully, `PRE_SPLIT_REGIONS` pre-spilts tables into the number of Regions as specified by `2^(PRE_SPLIT_REGIONS)`.

Note:

The value of `PRE_SPLIT_REGIONS` must be less than or equal to that of `SHARD_ROW_ID_BITS`.

The `tidb_scatter_region` global variable affects the behavior of `PRE_SPLIT_REGIONS` ↪ . This variable controls whether to wait for Regions to be pre-split and scattered before returning results after the table creation. If there are intensive writes after creating the table, you need to set the value of this variable to `1`, then TiDB will not return the results to the client until all the Regions are split and scattered. Otherwise, TiDB writes the data before the scattering is completed, which will have a significant impact on write performance.

Example

```
create table t (a int, b int,index idx1(a)) shard_row_id_bits = 4
    ↪ pre_split_regions=2;
```

After building the table, this statement splits `4 + 1` Regions for table t. `4 (2^2)` Regions are used to save table row data, and 1 Region is for saving the index data of `idx1`.

The ranges of the 4 table Regions are as follows:

```
region1:  [ -inf     ,  1<<61 )
region2:  [ 1<<61    ,  2<<61 )
region3:  [ 2<<61    ,  3<<61 )
region4:  [ 3<<61    ,  +inf  )
```

4.1.5.85.3 MySQL compatibility

This statement is a TiDB extension to MySQL syntax.

4.1.5.85.4 See also

There are two `SPLIT REGION` related session variables: `tidb_scatter_region`, `tidb_wait_split_region_finish` and `tidb_wait_split_region_timeout`. For details, see TiDB specific system variables and syntax.

4.1.5.86 START TRANSACTION

This statement starts a new transaction inside of TiDB. It is similar to the statements `BEGIN` and `SET autocommit=0`.

In the absence of a `START TRANSACTION` statement, every statement will by default autocommit in its own transaction. This behavior ensures MySQL compatibility.

4.1.5.86.1 Synopsis

BeginTransactionStmt:

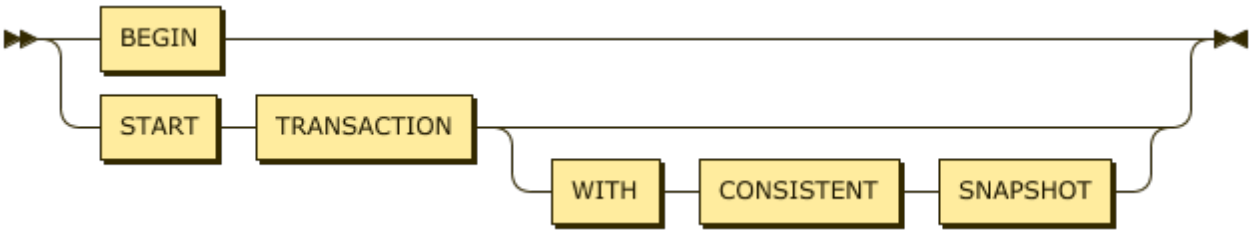

Figure 251: BeginTransactionStmt

4.1.5.86.2 Examples

```
mysql> CREATE TABLE t1 (a int NOT NULL PRIMARY KEY);
Query OK, 0 rows affected (0.12 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t1 VALUES (1);
Query OK, 1 row affected (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)
```

4.1.5.86.3 MySQL compatibility

This statement is understood to be partly compatible with MySQL.

- `START TRANSACTION` immediately starts a transaction inside TiDB. This differs from MySQL, where `START TRANSACTION` lazily creates a transaction unless the modifier `START TRANSACTION WITH CONSISTENT SNAPSHOT` is used.
- `READ ONLY` and its extended options are only syntactically compatible, and its effect is equivalent to `START TRANSACTION`.

Any compatibility differences should be reported via an issue on GitHub.

4.1.5.86.4 See also

- COMMIT
- ROLLBACK
- BEGIN

4.1.5.87 TRACE

The `TRACE` statement provides detailed information about query execution. It is intended to be viewed through a Graphical interface exposed by the TiDB server's status port.

4.1.5.87.1 Synopsis

TraceStmt:

Figure 252: TraceStmt

TraceableStmt:

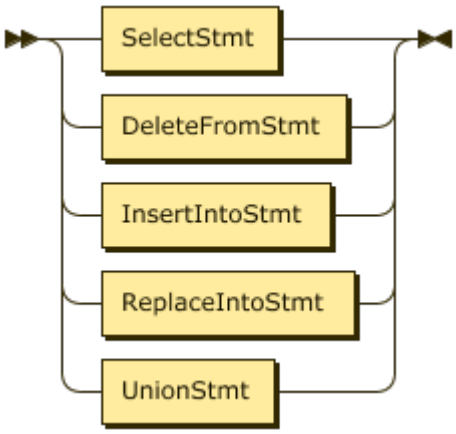

Figure 253: TraceableStmt

4.1.5.87.2 Examples

```
mysql> trace format='row' select * from mysql.user;
+---------------------------+-----------------+------------+
| operation                 | startTS         | duration   |
+---------------------------+-----------------+------------+
| session.getTxnFuture      | 10:33:34.647148 | 3.847µs    |
|   ─session.Execute        | 10:33:34.647146 | 536.233µs  |
|   ─session.ParseSQL       | 10:33:34.647182 | 19.868µs   |
|   ─executor.Compile       | 10:33:34.647219 | 295.688µs  |
|   ─session.runStmt        | 10:33:34.647533 | 116.229µs  |
|   ─session.CommitTxn      | 10:33:34.647631 | 5.44µs     |
|   ─recordSet.Next         | 10:33:34.647707 | 833.103µs  |
|   ─tableReader.Next       | 10:33:34.647709 | 806.783µs  |
|   ─recordSet.Next         | 10:33:34.648572 | 19.367µs   |
|   ─tableReader.Next       | 10:33:34.648575 | 1.783µs    |
```

```
+--------------------------+----------------+-----------+
10 rows in set (0.00 sec)

mysql> CREATE TABLE t1 (id int not null primary key AUTO_INCREMENT);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> TRACE FORMAT='json' SELECT * FROM t1 WHERE id = 2\G
operation: [
    {"ID":{"Trace":"60d20d005593de87","Span":"44e5b309242ffe2f","Parent":"79
        ↪ d146dac9a29a7e"},
        "Annotations":[
            {"Key":"Name","Value":"c2Vzc2lvbi5nZXRUeG5GdXR1cmU="},
            {"Key":"_schema:name","Value":null},
            {"Key":"Span.Start","Value":"
                ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE2MTQ3ODYtMDY6MDA="},
            {"Key":"Span.End","Value":"
                ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE2MjA0MDYtMDY6MDA="},
            {"Key":"_schema:Timespan","Value":null}
        ],
        "Sub":[
            {"ID":{"Trace":"60d20d005593de87","Span":"4dbf8f2ca373b4b0","
                ↪ Parent":"79d146dac9a29a7e"},
            "Annotations":[
                {"Key":"Name","Value":"c2Vzc2lvbi5QYXJzZVNRTA=="},
                {"Key":"_schema:name","Value":null},
                {"Key":"Span.Start","Value":"
                    ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE2NjE1MTQtMDY6MDA="},
                {"Key":"Span.End","Value":"
                    ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE3MDYxNjgtMDY6MDA="},
                {"Key":"_schema:Timespan","Value":null}
            ],
            "Sub":null},
            {"ID":{"Trace":"60d20d005593de87","Span":"6b6d6916df809604","
                ↪ Parent":"79d146dac9a29a7e"},
            "Annotations":[
                {"Key":"Name","Value":"ZXhlY3V0b3IuQ29tcGlsZQ=="},
                {"Key":"_schema:name","Value":null},
                {"Key":"Span.End","Value":"
                    ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE3NTcyODUtMDY6MDA="},
                {"Key":"Span.Start","Value":"
                    ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE3MzE0MjYtMDY6MDA="},
```

```
        {"Key":"_schema:Timespan","Value":null}
    ],
    "Sub":null},
    {"ID":{"Trace":"60d20d005593de87","Span":"3f1bcdd402a72911","
        ↪ Parent":"79d146dac9a29a7e"},
    "Annotations":[
        {"Key":"Name","Value":"c2Vzc2lvbi5Db21taXRUeG4="},
        {"Key":"_schema:name","Value":null},
        {"Key":"Span.Start","Value":"
            ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE3OTgyNjItMDY6MDA="},
        {"Key":"Span.End","Value":"
            ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE4MDU1NzYtMDY6MDA="},
        {"Key":"_schema:Timespan","Value":null}
    ],
    "Sub":null},
    {"ID":{"Trace":"60d20d005593de87","Span":"58c1f7d66dc5afbc","
        ↪ Parent":"79d146dac9a29a7e"},
    "Annotations":[
        {"Key":"Name","Value":"c2Vzc2lvbi5ydW5TdG10"},
        {"Key":"_schema:name","Value":null},
        {"Key":"Msg","Value":"
            ↪ eyJzcWwiOiJTRUxFQ1QgKiBGUk9NIHQxIFdIRVJFIGlkID0gMiJ9"},
        {"Key":"Time","Value":"
            ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE3ODA1NjgtMDY6MDA="},
        {"Key":"_schema:log","Value":null},
        {"Key":"Span.End","Value":"
            ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE4MTk5MzMtMDY6MDA="},
        {"Key":"Span.Start","Value":"
            ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE3NzcyNDItMDY6MDA="},
        {"Key":"_schema:Timespan","Value":null}
    ],
    "Sub":null},
    {"ID":{"Trace":"60d20d005593de87","Span":"6bd8cc440fb31ed7","
        ↪ Parent":"79d146dac9a29a7e"},
    "Annotations":[
        {"Key":"Name","Value":"c2Vzc2lvbi5FeGVjdXRl"},
        {"Key":"_schema:name","Value":null},
        {"Key":"Span.Start","Value":"
            ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE2MTEwODktMDY6MDA="},
        {"Key":"Span.End","Value":"
            ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE4NTU0My0wNjowMA=="},
        {"Key":"_schema:Timespan","Value":null}
    ],
    "Sub":null},
    {"ID":{"Trace":"60d20d005593de87","Span":"61d0b809f6cc018b","
```

```
            ↪ Parent":"79d146dac9a29a7e"},
         "Annotations":[
            {"Key":"Name","Value":"cmVjb3JkU2V0Lk5leHQ="},
            {"Key":"_schema:name","Value":null},
            {"Key":"Span.Start","Value":"
               ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDE4NzQ1NTYtMDY6MDA="},
            {"Key":"Span.End","Value":"
               ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDIyOTg4NjYtMDY6MDA="},
            {"Key":"_schema:Timespan","Value":null}
         ],
         "Sub":null},
         {"ID":{"Trace":"60d20d005593de87","Span":"2bd2c3d47ccb1133","
            ↪ Parent":"79d146dac9a29a7e"},
         "Annotations":[
            {"Key":"Name","Value":"cmVjb3JkU2V0Lk5leHQ="},
            {"Key":"_schema:name","Value":null},
            {"Key":"Span.Start","Value":"
               ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDIzMjY0ODgtMDY6MDA="},
            {"Key":"Span.End","Value":"
               ↪ MjAxOS0wNC0xN1QxMDozOToxMC45NDIzMjkwMDMtMDY6MDA="},
            {"Key":"_schema:Timespan","Value":null}
         ],
         "Sub":null}
      ]
   }
]
1 row in set (0.00 sec)
```

The JSON formatted trace can be pasted into the trace viewer, which is accessed via the TiDB status port:

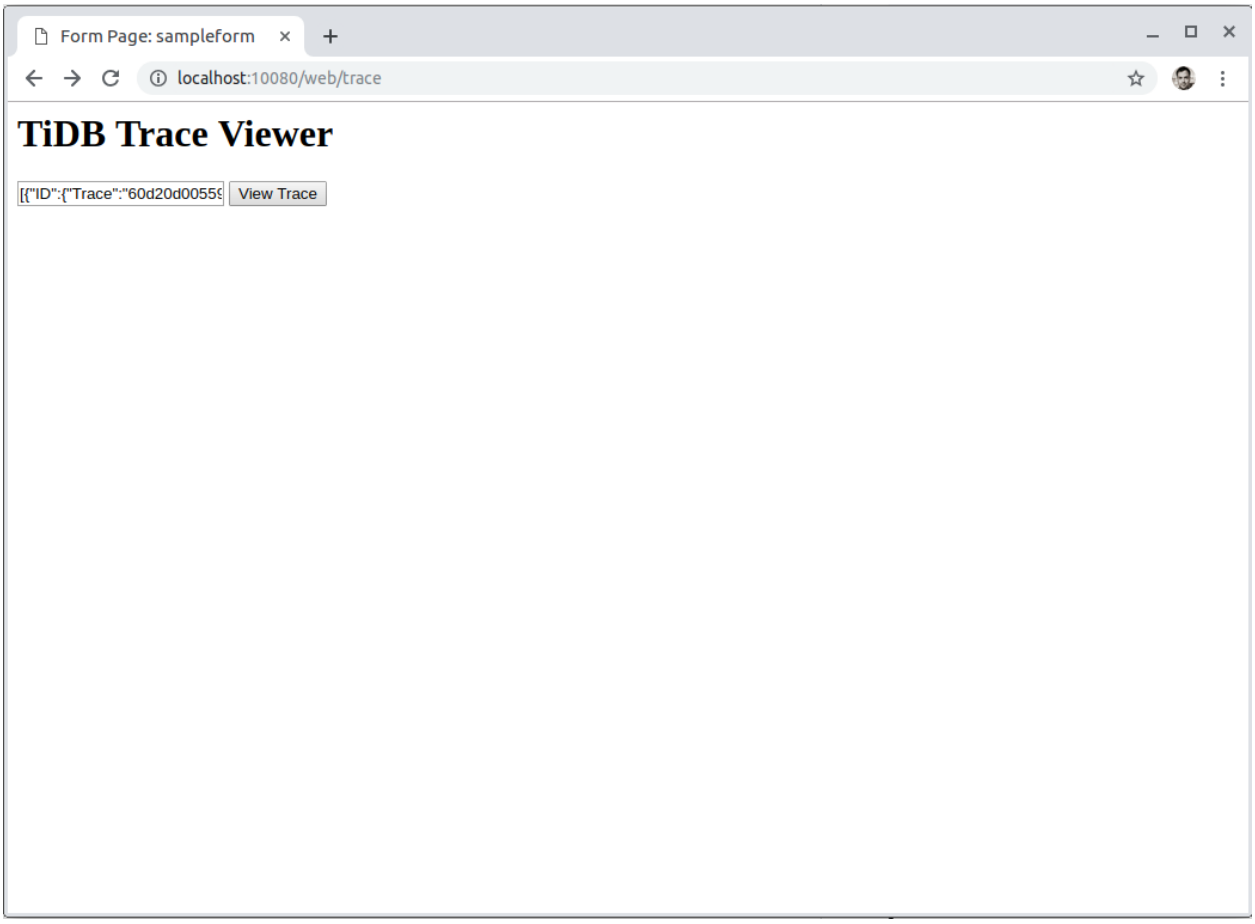

Figure 254: TiDB Trace Viewer-1

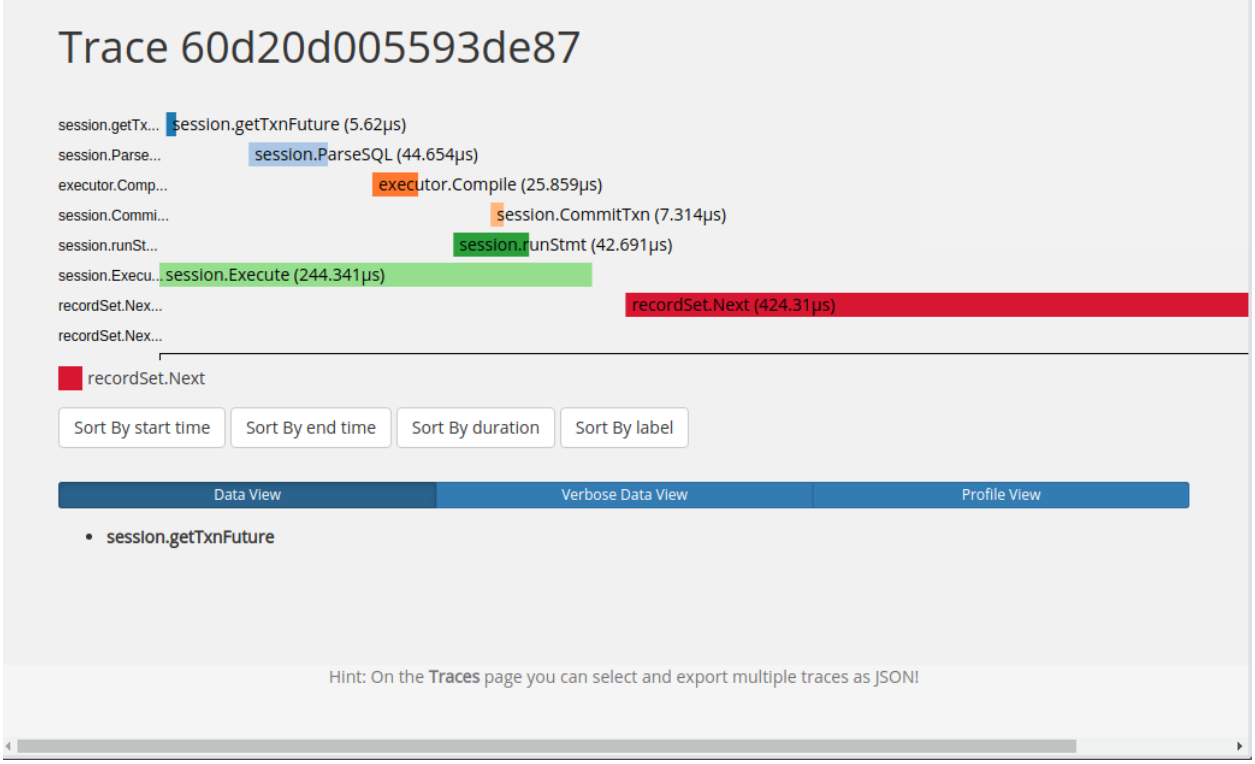

Figure 255: TiDB Trace Viewer-2

4.1.5.87.3 MySQL compatibility

This statement is a TiDB extension to MySQL syntax.

4.1.5.87.4 See also

- EXPLAIN ANALYZE

4.1.5.88 TRUNCATE

The `TRUNCATE` statement removes all data from the table in a non-transactional way. `TRUNCATE` can be thought of as semantically the same as `DROP TABLE` + `CREATE TABLE` with the previous definition.

Both `TRUNCATE TABLE tableName` and `TRUNCATE tableName` are valid syntax.

4.1.5.88.1 Synopsis

TruncateTableStmt:

Figure 256: TruncateTableStmt

OptTable:

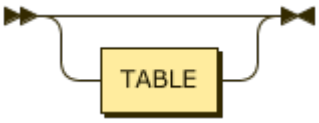

Figure 257: OptTable

TableName:

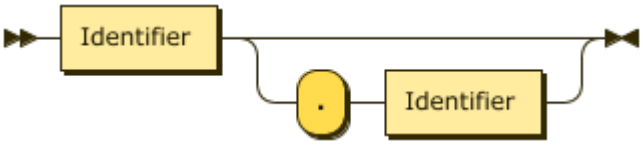

Figure 258: TableName

4.1.5.88.2 Examples

```
mysql> CREATE TABLE t1 (a INT NOT NULL PRIMARY KEY);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.01 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM t1;
+---+
| a |
+---+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
+---+
5 rows in set (0.00 sec)

mysql> TRUNCATE t1;
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> SELECT * FROM t1;
Empty set (0.00 sec)

mysql> INSERT INTO t1 VALUES (1),(2),(3),(4),(5);
Query OK, 5 rows affected (0.01 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> TRUNCATE TABLE t1;
Query OK, 0 rows affected (0.11 sec)
```

4.1.5.88.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.88.4 See also

- DROP TABLE
- DELETE
- CREATE TABLE
- SHOW CREATE TABLE

4.1.5.89 UPDATE

The `UPDATE` statement is used to modify data in a specified table.

4.1.5.89.1 Synopsis

UpdateStmt:

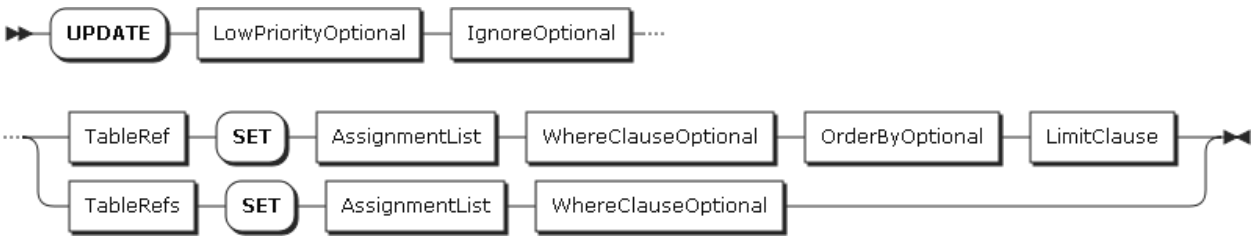

Figure 259: UpdateStmt

TableRef:

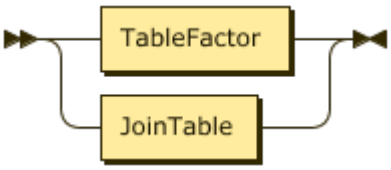

Figure 260: TableRef

TableRefs:

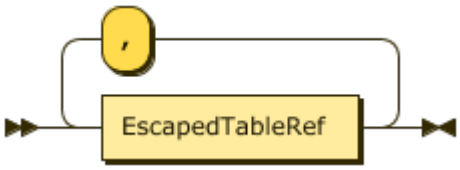

Figure 261: TableRefs

AssignmentList:

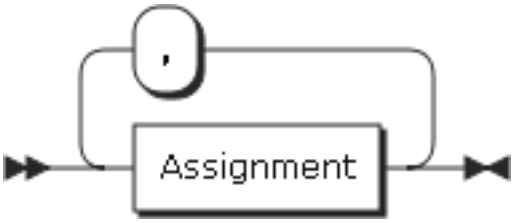

Figure 262: AssignmentList

WhereClauseOptional:

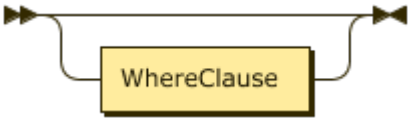

Figure 263: WhereClauseOptional

4.1.5.89.2 Examples

```
mysql> CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, c1 INT
    ↪ NOT NULL);
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO t1 (c1) VALUES (1), (2), (3);
Query OK, 3 rows affected (0.02 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> SELECT * FROM t1;
+----+----+
| id | c1 |
+----+----+
|  1 |  1 |
|  2 |  2 |
|  3 |  3 |
+----+----+
3 rows in set (0.00 sec)

mysql> UPDATE t1 SET c1=5 WHERE c1=3;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> SELECT * FROM t1;
+----+----+
| id | c1 |
+----+----+
|  1 |  1 |
|  2 |  2 |
|  3 |  5 |
+----+----+
3 rows in set (0.00 sec)
```

4.1.5.89.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.89.4 See also

- INSERT
- SELECT
- DELETE
- REPLACE

4.1.5.90 USE

The `USE` statement selects a current database for the user session.

4.1.5.90.1 Synopsis

UseStmt:

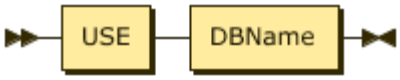

Figure 264: UseStmt

DBName:

Figure 265: DBName

4.1.5.90.2 Examples

```
mysql> USE mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+----------------------+
| Tables_in_mysql      |
+----------------------+
| GLOBAL_VARIABLES     |
| bind_info            |
| columns_priv         |
| db                   |
| default_roles        |
| gc_delete_range      |
| gc_delete_range_done |
| help_topic           |
| role_edges           |
| stats_buckets        |
| stats_feedback       |
| stats_histograms     |
| stats_meta           |
| tables_priv          |
| tidb                 |
| user                 |
+----------------------+
16 rows in set (0.00 sec)

mysql> CREATE DATABASE newtest;
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> USE newtest;
Database changed
mysql> SHOW TABLES;
Empty set (0.00 sec)

mysql> CREATE TABLE t1 (a int);
Query OK, 0 rows affected (0.10 sec)

mysql> SHOW TABLES;
+------------------+
| Tables_in_newtest |
+------------------+
| t1               |
+------------------+
1 row in set (0.00 sec)
```

4.1.5.90.3 MySQL compatibility

This statement is understood to be fully compatible with MySQL. Any compatibility differences should be reported via an issue on GitHub.

4.1.5.90.4 See also

- CREATE DATABASE
- SHOW TABLES

4.1.6 Constraints

TiDB supports almost the same constraints as MySQL.

4.1.6.1 NOT NULL

NOT NULL constraints supported by TiDB are the same as those supported by MySQL.

For example:

```
CREATE TABLE users (
 id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
 age INT NOT NULL,
 last_login TIMESTAMP
);
```

```
INSERT INTO users (id,age,last_login) VALUES (NULL,123,NOW());
```

```
Query OK, 1 row affected (0.02 sec)
```

```
INSERT INTO users (id,age,last_login) VALUES (NULL,NULL,NOW());
```

```
ERROR 1048 (23000): Column 'age' cannot be null
```

```
INSERT INTO users (id,age,last_login) VALUES (NULL,123,NULL);
```

```
Query OK, 1 row affected (0.03 sec)
```

- The first `INSERT` statement succeeds because it is possible to assign `NULL` to the `AUTO_INCREMENT` column. TiDB generates sequence numbers automatically.
- The second `INSERT` statement fails because the `age` column is defined as `NOT NULL`.
- The third `INSERT` statement succeeds because the `last_login` column is not explicitly defined as `NOT NULL`. NULL values are allowed by default.

4.1.6.2 CHECK

TiDB parses but ignores `CHECK` constraints. This is MySQL 5.7 compatible behavior.

For example:

```
DROP TABLE IF EXISTS users;
CREATE TABLE users (
 id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
 username VARCHAR(60) NOT NULL,
 UNIQUE KEY (username),
 CONSTRAINT min_username_length CHECK (CHARACTER_LENGTH(username) >=4)
);
INSERT INTO users (username) VALUES ('a');
SELECT * FROM users;
```

4.1.6.3 UNIQUE KEY

Depending on the transaction mode and the value of `tidb_constraint_check_in_place` ↪ , TiDB might check `UNIQUE` constraints lazily. This helps improve performance by batching network access.

For example:

```
DROP TABLE IF EXISTS users;
CREATE TABLE users (
 id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
 username VARCHAR(60) NOT NULL,
 UNIQUE KEY (username)
```

```
);
INSERT INTO users (username) VALUES ('dave'), ('sarah'), ('bill');
```

With the default of pessimistic locking:

```
BEGIN;
INSERT INTO users (username) VALUES ('jane'), ('chris'), ('bill');
```

```
ERROR 1062 (23000): Duplicate entry 'bill' for key 'username'
```

With optimistic locking and `tidb_constraint_check_in_place=0`:

```
BEGIN OPTIMISTIC;
INSERT INTO users (username) VALUES ('jane'), ('chris'), ('bill');
```

```
Query OK, 0 rows affected (0.00 sec)

Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
INSERT INTO users (username) VALUES ('steve'),('elizabeth');
```

```
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

```
COMMIT;
```

```
ERROR 1062 (23000): Duplicate entry 'bill' for key 'username'
```

In the optimistic example, the unique check was delayed until the transaction is committed. This resulted in a duplicate key error, because the value `bill` was already present.

You can disable this behavior by setting `tidb_constraint_check_in_place` to `1`. This variable setting does not take effect on pessimistic transactions, because in the pessimistic transaction mode the constraints are always checked when the statement is executed. When `tidb_constraint_check_in_place=1`, the unique constraint is checked when the statement is executed.

For example:

```
DROP TABLE IF EXISTS users;
CREATE TABLE users (
 id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
 username VARCHAR(60) NOT NULL,
 UNIQUE KEY (username)
);
INSERT INTO users (username) VALUES ('dave'), ('sarah'), ('bill');
```

```
SET tidb_constraint_check_in_place = 1;
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
BEGIN OPTIMISTIC;
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
INSERT INTO users (username) VALUES ('jane'), ('chris'), ('bill');
```

```
ERROR 1062 (23000): Duplicate entry 'bill' for key 'username'
..
```

The first `INSERT` statement caused a duplicate key error. This causes additional network communication overhead and might reduce the throughput of `INSERT` operations.

4.1.6.4 PRIMARY KEY

Like MySQL, primary key constraints contain unique constraints, that is, creating a primary key constraint is equivalent to having a unique constraint. In addition, other primary key constraints of TiDB are also similar to those of MySQL.

For example:

```
CREATE TABLE t1 (a INT NOT NULL PRIMARY KEY);
```

```
Query OK, 0 rows affected (0.12 sec)
```

```
CREATE TABLE t2 (a INT NULL PRIMARY KEY);
```

```
ERROR 1171 (42000): All parts of a PRIMARY KEY must be NOT NULL; if you need
    ↪  NULL in a key, use UNIQUE instead
```

```
CREATE TABLE t3 (a INT NOT NULL PRIMARY KEY, b INT NOT NULL PRIMARY KEY);
```

```
ERROR 1068 (42000): Multiple primary key defined
```

```
CREATE TABLE t4 (a INT NOT NULL, b INT NOT NULL, PRIMARY KEY (a,b));
```

```
Query OK, 0 rows affected (0.10 sec)
```

- Table `t2` failed to be created, because column `a` is defined as the primary key and does not allow NULL values.

- Table `t3` failed to be created, because a table can only have one primary key.
- Table `t4` was created successfully, because even though there can be only one primary key, TiDB supports defining multiple columns as the composite primary key.

In addition to the rules above, by default, TiDB has an additional restriction that once a table is successfully created, its primary key cannot be changed. If you need to add or remove the primary key, set `alter-primary-key` to `true` in the TiDB configuration file, and restart the TiDB instance to make it effective.

When the feature of adding or deleting the primary key is enabled, TiDB allows adding the primary key to or deleting the primary key from the table. However, if a table with an integer type primary key has been created before the feature is enabled, you cannot delete its primary key constraint even when you enable the adding or deleting primary key feature.

4.1.6.5 FOREIGN KEY

> **Note:**
>
> TiDB has limited support for foreign key constraints.

TiDB supports creating `FOREIGN KEY` constraints in DDL commands.

For example:

```
CREATE TABLE users (
 id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
 doc JSON
);
CREATE TABLE orders (
 id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
 user_id INT NOT NULL,
 doc JSON,
 FOREIGN KEY fk_user_id (user_id) REFERENCES users(id)
);
```

```
SELECT table_name, column_name, constraint_name, referenced_table_name,
    ↪ referenced_column_name
FROM information_schema.key_column_usage WHERE table_name IN ('users', '
    ↪ orders');
```

```
+--
    ↪ ----------+-------------+-----------------+----------------------+-----------
    ↪
```

```
| table_name | column_name | constraint_name | referenced_table_name |
    ↪ referenced_column_name |
+--
    ↪ ----------+-------------+-----------------+-----------------------+-----------
    ↪
| users      | id          | PRIMARY         | NULL                  | NULL
    ↪                  |
| orders     | id          | PRIMARY         | NULL                  | NULL
    ↪                  |
| orders     | user_id     | fk_user_id      | users                 | id
    ↪                    |
+--
    ↪ ----------+-------------+-----------------+-----------------------+-----------
    ↪
3 rows in set (0.00 sec)
```

TiDB also supports the `DROP FOREIGN KEY` and `ADD FOREIGN KEY` syntax via the `ALTER TABLE` command.

```
ALTER TABLE orders DROP FOREIGN KEY fk_user_id;
ALTER TABLE orders ADD FOREIGN KEY fk_user_id (user_id) REFERENCES users(id
    ↪ );
```

4.1.6.5.1 Notes

- TiDB supports foreign keys to avoid errors caused by this syntax when you migrate data from other databases to TiDB.

  However, TiDB does not perform constraint checking on foreign keys in DML statements. For example, even if there is no record with id=123 in the users table, the following transactions can be submitted successfully.

  ```
  START TRANSACTION;
  INSERT INTO orders (user_id, doc) VALUES (123, NULL);
  COMMIT;
  ```

4.1.7 Generated Columns

> **Warning:**
>
> This is still an experimental feature. It is recommended **not** to use this feature in the production environment.

TiDB supports generated columns as part of MySQL 5.7 compatibility. One of the primary use cases for generated columns is to extract data out of a JSON data type and enable it to be indexed.

4.1.7.1 Index JSON using generated column

In both MySQL 5.7 and TiDB, columns of type JSON can not be indexed directly. i.e. The following table structure is **not supported**:

```
CREATE TABLE person (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    address_info JSON,
    KEY (address_info)
);
```

To index a JSON column, you must first extract it as a generated column.

Using the `city` stored generated column as an example, you are then able to add an index:

```
CREATE TABLE person (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    address_info JSON,
    city VARCHAR(64) AS (JSON_UNQUOTE(JSON_EXTRACT(address_info, '$.city')))
        ↪ STORED,
    KEY (city)
);
```

In this table, the `city` column is a **generated column**. As the name implies, the column is generated from other columns in the table, and cannot be assigned a value when inserted or updated. This column is generated based on a defined expression and is stored in the database. Thus this column can be read directly, not in a way that its dependent column `address_info` is read first and then the data is calculated. The index on `city` however is *stored* and uses the same structure as other indexes of the type `varchar(64)`.

You can use the index on the stored generated column in order to speed up the following statement:

```
SELECT name, id FROM person WHERE city = 'Beijing';
```

If no data exists at path `$.city`, `JSON_EXTRACT` returns `NULL`. If you want to enforce a constraint that `city` must be `NOT NULL`, you can define the virtual column as follows:

```
CREATE TABLE person (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
```

```
    address_info JSON,
    city VARCHAR(64) AS (JSON_UNQUOTE(JSON_EXTRACT(address_info, '$.city')))
        ↪ STORED NOT NULL,
    KEY (city)
);
```

Both `INSERT` and `UPDATE` statements check virtual column definitions. Rows that do not pass validation return errors:

```
mysql> INSERT INTO person (name, address_info) VALUES ('Morgan',
    ↪ JSON_OBJECT('Country', 'Canada'));
ERROR 1048 (23000): Column 'city' cannot be null
```

4.1.7.2 Use generated virtual columns

TiDB also supports generated virtual columns. Different from generated store columns, generated virtual columns are **virtual** in that they are generated as needed and are not stored in the database or cached in the memory.

```
CREATE TABLE person (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    address_info JSON,
    city VARCHAR(64) AS (JSON_UNQUOTE(JSON_EXTRACT(address_info, '$.city')))
        ↪ VIRTUAL
);
```

4.1.7.3 Limitations

The current limitations of JSON and generated columns are as follows:

- You cannot add the generated column in the storage type of `STORED` through `ALTER` ↪ `TABLE`.
- You cannot create an index on the generated column through `ALTER TABLE`.
- You can neither convert a generated stored column to a normal column through the `ALTER TABLE` statement nor convert a normal column to a generated stored column.
- You cannot modify the **expression** of a generated stored column through the `ALTER` ↪ `TABLE` statement.
- Not all JSON functions are supported.

4.1.8 Partitioning

This document introduces TiDB's implementation of partitioning.

4.1.8.1 Partitioning types

This section introduces the types of partitioning which are available in TiDB. Currently, TiDB supports range partitioning and hash partitioning.

Range partitioning is used to resolve the performance issues caused by a large amount of deletions in the application, and it supports fast drop partition operations. Hash partitioning is used to scatter the data when there are a large amount of writes.

4.1.8.1.1 Range partitioning

When a table is partitioned by range, each partition contains rows for which the partitioning expression value lies within a given range. Ranges have to be contiguous but not overlapping. You can define it by using the `VALUES LESS THAN` operation.

Assume you need to create a table that contains personnel records as follows:

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
);
```

You can partition a table by range in various ways as needed. For example, you can partition it by using the `store_id` column:

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
)

PARTITION BY RANGE (store_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN (21)
);
```

In this partition scheme, all rows corresponding to employees whose `store_id` is 1 through 5 are stored in the `p0` partition while all employees whose `store_id` is 6 through 10 are stored in `p1`. Range partitioning requires the partitions to be ordered, from lowest to highest.

If you insert a row of data `(72, 'Mitchell', 'Wilson', '1998-06-25', NULL, 13)`, it falls in the `p2` partition. But if you insert a record whose `store_id` is larger than 20, an error is reported because TiDB can not know which partition this record should be inserted into. In this case, you can use `MAXVALUE` when creating a table:

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
)

PARTITION BY RANGE (store_id) (
    PARTITION p0 VALUES LESS THAN (6),
    PARTITION p1 VALUES LESS THAN (11),
    PARTITION p2 VALUES LESS THAN (16),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

`MAXVALUE` represents an integer value that is larger than all other integer values. Now, all records whose `store_id` is equal to or larger than 16 (the highest value defined) are stored in the `p3` partition.

You can also partition a table by employees' job codes, which are the values of the `job_code` column. Assume that two-digit job codes stand for regular employees, three-digit codes stand for office and customer support personnel, and four-digit codes stand for managerial personnel. Then you can create a partitioned table like this:

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT NOT NULL,
    store_id INT NOT NULL
)

PARTITION BY RANGE (job_code) (
```

```
    PARTITION p0 VALUES LESS THAN (100),
    PARTITION p1 VALUES LESS THAN (1000),
    PARTITION p2 VALUES LESS THAN (10000)
);
```

In this example, all rows relating to regular employees are stored in the `p0` partition, all office and customer support personnel in the `p1` partition, and all managerial personnel in the `p2` partition.

Besides splitting up the table by `store_id`, you can also partition a table by dates. For example, you can partition by employees' separation year:

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)

PARTITION BY RANGE ( YEAR(separated) ) (
    PARTITION p0 VALUES LESS THAN (1991),
    PARTITION p1 VALUES LESS THAN (1996),
    PARTITION p2 VALUES LESS THAN (2001),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

In range partitioning, you can partition based on the values of the `timestamp` column and use the `unix_timestamp()` function, for example:

```
CREATE TABLE quarterly_report_status (
    report_id INT NOT NULL,
    report_status VARCHAR(20) NOT NULL,
    report_updated TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
        ↪ CURRENT_TIMESTAMP
)

PARTITION BY RANGE ( UNIX_TIMESTAMP(report_updated) ) (
    PARTITION p0 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-01-01 00:00:00') ),
    PARTITION p1 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-04-01 00:00:00') ),
    PARTITION p2 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-07-01 00:00:00') ),
    PARTITION p3 VALUES LESS THAN ( UNIX_TIMESTAMP('2008-10-01 00:00:00') ),
    PARTITION p4 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-01-01 00:00:00') ),
    PARTITION p5 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-04-01 00:00:00') ),
```

```
    PARTITION p6 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-07-01 00:00:00') ),
    PARTITION p7 VALUES LESS THAN ( UNIX_TIMESTAMP('2009-10-01 00:00:00') ),
    PARTITION p8 VALUES LESS THAN ( UNIX_TIMESTAMP('2010-01-01 00:00:00') ),
    PARTITION p9 VALUES LESS THAN (MAXVALUE)
);
```

It is not allowed to use any other partitioning expression that contains the timestamp values.

Range partitioning is particularly useful when one or more of the following conditions are satisfied:

- You want to delete the old data. If you use the `employees` table in the previous example, you can delete all records of employees who left this company before the year 1991 by simply using `ALTER TABLE employees DROP PARTITION p0;`. It is faster than executing the `DELETE FROM employees WHERE YEAR(separated)<= 1990;` operation.
- You want to use a column that contains time or date values, or containing values arising from some other series.
- You need to frequently run queries on the columns used for partitioning. For example, when executing a query like `EXPLAIN SELECT COUNT(*)FROM employees WHERE ↪ separated BETWEEN '2000-01-01' AND '2000-12-31' GROUP BY store_id;`, TiDB can quickly know that only the data in the `p2` partition needs to be scanned, because the other partitions do not match the `WHERE` condition.

4.1.8.1.2 Hash partitioning

Hash partitioning is used to make sure that data is evenly scattered into a certain number of partitions. With range partitioning, you must specify the range of the column values for each partition when you use range partitioning, while you just need to specify the number of partitions when you use hash partitioning.

Partitioning by hash requires you to append a `PARTITION BY HASH (expr)` clause to the `CREATE TABLE` statement. `expr` is an expression that returns an integer. It can be a column name if the type of this column is integer. In addition, you might also need to append `PARTITIONS num`, where `num` is a positive integer indicating how many partitions a table is divided into.

The following operation creates a hash partitioned table, which is divided into 4 partitions by `store_id`:

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
```

```
    store_id INT
)

PARTITION BY HASH(store_id)
PARTITIONS 4;
```

If `PARTITIONS num` is not specified, the default number of partitions is 1.

You can also use an SQL expression that returns an integer for `expr`. For example, you can partition a table by the hire year:

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)

PARTITION BY HASH( YEAR(hired) )
PARTITIONS 4;
```

The most efficient hash function is one which operates upon a single table column, and whose value increases or decreases consistently with the column value, as this allows for "pruning" on ranges of partitions.

For example, `date_col` is a column whose type is `DATE`, and the value of the `TO_DAYS (date_col)` expression varies with the value of `date_col`. `YEAR(date_col)` is different from `TO_DAYS(date_col)`, because not every possible change in `date_col` produces an equivalent change in `YEAR(date_col)`. Even so, `YEAR(date_col)` is still a good hash function, because its value varies in proportion to the value of `date_col`.

In contrast, assume that you have an `int_col` column whose type is `INT`. Now consider about the expression `POW(5-int_col,3)+ 6`. It is not a good hash function though, because as the value of `int_col` changes, the result of the expression does not change proportionally. A value change in `int_col` might result in a huge change in the expression result. For example, when `int_col` changes from 5 to 6, the change of the expression result is -1. But the result change might be -7 when `int_col` changes from 6 to 7.

In conclusion, when the expression has a form that is closer to `y = cx`, it is more suitable to be a hash function. Because the more non-linear an expression is, the more unevenly scattered the data among the partitions tends to be.

In theory, pruning is also possible for expressions involving more than one column value, but determining which of such expressions are suitable can be quite difficult and time-consuming. For this reason, the use of hashing expressions involving multiple columns is not particularly recommended.

When using `PARTITIION BY HASH`, TiDB decides which partition the data should fall into based on the modulus of the result of the expression. In other words, if a partitioning expression is `expr` and the number of partitions is `num`, `MOD(expr, num)` decides the partition in which the data is stored. Assume that `t1` is defined as follows:

```
CREATE TABLE t1 (col1 INT, col2 CHAR(5), col3 DATE)
    PARTITION BY HASH( YEAR(col3) )
    PARTITIONS 4;
```

When you insert a row of data into `t1` and the value of `col3` is '2005-09-15', then this row is inserted into partition 1:

```
MOD(YEAR('2005-09-01'),4)
=  MOD(2005,4)
=  1
```

4.1.8.1.3 How TiDB partitioning handles NULL

It is allowed in TiDB to use `NULL` as the calculation result of a partitioning expression.

> **Note:**
>
> `NULL` is not an integer. TiDB's partitioning implementation treats `NULL` as being less than any other integer values, just as `ORDER BY` does.

Handling of NULL with range partitioning

When you insert a row into a table partitioned by range, and the column value used to determine the partition is `NULL`, then this row is inserted into the lowest partition.

```
CREATE TABLE t1 (
    c1 INT,
    c2 VARCHAR(20)
)

PARTITION BY RANGE(c1) (
    PARTITION p0 VALUES LESS THAN (0),
    PARTITION p1 VALUES LESS THAN (10),
    PARTITION p2 VALUES LESS THAN MAXVALUE
);
```

```
Query OK, 0 rows affected (0.09 sec)
```

```
select * from t1 partition(p0);
```

```
+------|--------+
| c1   | c2     |
+------|--------+
| NULL | mothra |
+------|--------+
1 row in set (0.00 sec)
```

```
select * from t1 partition(p1);
```

```
Empty set (0.00 sec)
```

```
select * from t1 partition(p2);
```

```
Empty set (0.00 sec)
```

Drop the p0 partition and verify the result:

```
alter table t1 drop partition p0;
```

```
Query OK, 0 rows affected (0.08 sec)
```

```
select * from t1;
```

```
Empty set (0.00 sec)
```

Handling of NULL with hash partitioning

When partitioning tables by hash, there is a different way of handling NULL value - if the calculation result of the partitioning expression is NULL, it is considered as 0.

```
CREATE TABLE th (
    c1 INT,
    c2 VARCHAR(20)
)

PARTITION BY HASH(c1)
PARTITIONS 2;
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
INSERT INTO th VALUES (NULL, 'mothra'), (0, 'gigan');
```

```
Query OK, 2 rows affected (0.04 sec)
```

```
select * from th partition (p0);
```

```
+------|--------+
| c1   | c2     |
+------|--------+
| NULL | mothra |
|    0 | gigan  |
+------|--------+
2 rows in set (0.00 sec)
```

```
select * from th partition (p1);
```

```
Empty set (0.00 sec)
```

You can see that the inserted record `(NULL, 'mothra')` falls into the same partition as `(0, 'gigan')`.

4.1.8.2 Partition management

You can add, drop, merge, split, redefine partitions by using `ALTER TABLE` statements.

4.1.8.2.1 Range partition management

Create a partitioned table:

```
CREATE TABLE members (
    id INT,
    fname VARCHAR(25),
    lname VARCHAR(25),
    dob DATE
)

PARTITION BY RANGE( YEAR(dob) ) (
    PARTITION p0 VALUES LESS THAN (1980),
    PARTITION p1 VALUES LESS THAN (1990),
    PARTITION p2 VALUES LESS THAN (2000)
);
```

Drop a partition:

```
ALTER TABLE members DROP PARTITION p2;
```

```
Query OK, 0 rows affected (0.03 sec)
```

Empty a partition:

```
ALTER TABLE members TRUNCATE PARTITION p1;
```

```
Query OK, 0 rows affected (0.03 sec)
```

> **Note:**
>
> `ALTER TABLE ... REORGANIZE PARTITION` is currently unsupported in TiDB.

Add a partition:

```
ALTER TABLE members ADD PARTITION (PARTITION p3 VALUES LESS THAN (2010));
```

When partitioning tables by range, `ADD PARTITION` can be only appended to the very end of a partition list. If it is appended to an existing partition range, an error is reported:

```
ALTER TABLE members
    ADD PARTITION (
    PARTITION n VALUES LESS THAN (1970));
```

```
ERROR 1463 (HY000): VALUES LESS THAN value must be strictly »
   increasing for each partition
```

4.1.8.2.2 Hash partition management

Unlike range partitioning, `DROP PARTITION` is not supported in hash partitioning.

Currently, `ALTER TABLE ... COALESCE PARTITION` is not supported in TiDB as well.

4.1.8.3 Partition pruning

Partition pruning is an optimization which is based on a very simple idea - do not scan the partitions that do not match.

Assume that you create a partitioned table `t1`:

```
CREATE TABLE t1 (
    fname VARCHAR(50) NOT NULL,
    lname VARCHAR(50) NOT NULL,
    region_code TINYINT UNSIGNED NOT NULL,
    dob DATE NOT NULL
)
```

```
PARTITION BY RANGE( region_code ) (
    PARTITION p0 VALUES LESS THAN (64),
    PARTITION p1 VALUES LESS THAN (128),
    PARTITION p2 VALUES LESS THAN (192),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);
```

If you want to get the result of this `SELECT` statement:

```
SELECT fname, lname, region_code, dob
    FROM t1
    WHERE region_code > 125 AND region_code < 130;
```

It is evident that the result falls in either the `p1` or the `p2` partition, that is, you just need to search for the matching rows in `p1` and `p2`. Excluding the unneeded partitions is so-called "pruning". If the optimizer is able to prune a part of partitions, the execution of the query in the partitioned table will be much faster than that in a non-partitioned table.

The optimizer can prune partitions through `WHERE` conditions in the following two scenarios:

- partition_column = constant
- partition_column IN (constant1, constant2, ..., constantN)

4.1.8.3.1 Some cases for partition pruning to take effect

1. Partition pruning uses the query conditions on the partitioned table, so if the query conditions can not be pushed down to the partitioned table according to the planner's optimization rules, partition pruning does not apply for this query.

   For example:

   ```
   create table t1 (x int) partition by range (x) (
          partition p0 values less than (5),
          partition p1 values less than (10));
   create table t2 (x int);
   ```

   ```
   explain select * from t1 left join t2 on t1.x = t2.x where t2.x > 5;
   ```

   In this query, the left out join is converted to the inner join, and then `t1.x > 5` is derived from `t1.x = t2.x` and `t2.x > 5`, so it could be used in partition pruning and only the partition `p1` remains.

   ```
   explain select * from t1 left join t2 on t1.x = t2.x and t2.x > 5;
   ```

In this query, `t2.x > 5` can not be pushed down to the `t1` partitioned table, so partition pruning would not take effect for this query.

2. Since partition pruning is done during the plan optimizing phase, it does not apply for those cases that filter conditions are unknown until the execution phase.

   For example:

```
create table t1 (x int) partition by range (x) (
       partition p0 values less than (5),
       partition p1 values less than (10));
```

```
explain select * from t2 where x < (select * from t1 where t2.x < t1.x
   ↪  and t2.x < 2);
```

   This query reads a row from `t2` and uses the result for the subquery on `t1`. Theoretically, partition pruning could benefit from `t1.x > val` expression in the subquery, but it does not take effect there as that happens in the execution phase.

3. As a result of a limitation from current implementation, if a query condition can not be pushed down to TiKV, it can not be used by the partition pruning.

   Take the `fn(col)` expression as an example. If the TiKV coprocessor supports this `fn` function, `fn(col)` may be pushed down to the the leaf node (that is, partitioned table) according to the predicate push-down rule during the plan optimizing phase, and partition pruning can use it.

   If the TiKV coprocessor does not support this `fn` function, `fn(col)` would not be pushed down to the leaf node. Instead, it becomes a `Selection` node above the leaf node. The current partition pruning implementation does not support this kind of plan tree.

4. For hash partition, the only query supported by partition pruning is the equal condition.

5. For range partition, for partition pruning to take effect, the partition expression must be in those forms: `col` or `fn(col)`, and the query condition must be one of `>`, `<`, `=`, `>=`, and `<=`. If the partition expression is in the form of `fn(col)`, the `fn` function must be monotonous.

   If the `fn` function is monotonous, for any `x` and `y`, if `x > y`, then `fn(x) > fn(y)`. Then this `fn` function can be called strictly monotonous. For any `x` and `y`, if `x > y`, then `fn(x) >= fn(y)`. In this case, `fn` could also be called "monotonous". In theory, all monotonous functions are supported by partition pruning.

   Currently, partition pruning in TiDB only support those monotonous functions:

```
unix_timestamp
to_days
```

   For example, the partition expression is a simple column:

```
create table t (id int) partition by range (id) (
       partition p0 values less than (5),
       partition p1 values less than (10));
select * from t where t > 6;
```

Or the partition expression is in the form of `fn(col)` where `fn` is `to_days`:

```
create table t (dt datetime) partition by range (to_days(id)) (
       partition p0 values less than (to_days('2020-04-01')),
       partition p1 values less than (to_days('2020-05-01')));
select * from t where t > '2020-04-18';
```

An exception is `floor(unix_timestamp())` as the partition expression. TiDB does some optimization for that case by case, so it is supported by partition pruning.

```
create table t (ts timestamp(3) not null default current_timestamp(3))
partition by range (floor(unix_timestamp(ts))) (
       partition p0 values less than (unix_timestamp('2020-04-01
          ↪ 00:00:00')),
       partition p1 values less than (unix_timestamp('2020-05-01
          ↪ 00:00:00')));
select * from t where t > '2020-04-18 02:00:42.123';
```

4.1.8.4 Partition selection

`SELECT` statements support partition selection, which is implemented by using a `PARTITION` option.

```
CREATE TABLE employees (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    fname VARCHAR(25) NOT NULL,
    lname VARCHAR(25) NOT NULL,
    store_id INT NOT NULL,
    department_id INT NOT NULL
)

PARTITION BY RANGE(id) (
    PARTITION p0 VALUES LESS THAN (5),
    PARTITION p1 VALUES LESS THAN (10),
    PARTITION p2 VALUES LESS THAN (15),
    PARTITION p3 VALUES LESS THAN MAXVALUE
);

INSERT INTO employees VALUES
    ('', 'Bob', 'Taylor', 3, 2), ('', 'Frank', 'Williams', 1, 2),
```

```
    ('', 'Ellen', 'Johnson', 3, 4), ('', 'Jim', 'Smith', 2, 4),
    ('', 'Mary', 'Jones', 1, 1), ('', 'Linda', 'Black', 2, 3),
    ('', 'Ed', 'Jones', 2, 1), ('', 'June', 'Wilson', 3, 1),
    ('', 'Andy', 'Smith', 1, 3), ('', 'Lou', 'Waters', 2, 4),
    ('', 'Jill', 'Stone', 1, 4), ('', 'Roger', 'White', 3, 2),
    ('', 'Howard', 'Andrews', 1, 2), ('', 'Fred', 'Goldberg', 3, 3),
    ('', 'Barbara', 'Brown', 2, 3), ('', 'Alice', 'Rogers', 2, 2),
    ('', 'Mark', 'Morgan', 3, 3), ('', 'Karen', 'Cole', 3, 2);
```

You can view the rows stored in the p1 partition:

```
SELECT * FROM employees PARTITION (p1);
```

```
+----|-------|--------|----------|---------------+
| id | fname | lname | store_id | department_id |
+----|-------|--------|----------|---------------+
|  5 | Mary | Jones |       1 |             1 |
|  6 | Linda | Black |       2 |             3 |
|  7 | Ed   | Jones |       2 |             1 |
|  8 | June | Wilson |       3 |             1 |
|  9 | Andy | Smith |       1 |             3 |
+----|-------|--------|----------|---------------+
5 rows in set (0.00 sec)
```

If you want to get the rows in multiple partitions, you can use a list of partition names which are separated by commas. For example, `SELECT * FROM employees PARTITION (p1 , p2)` returns all rows in the p1 and p2 partitions.

When you use partition selection, you can still use `WHERE` conditions and options such as `ORDER BY` and `LIMIT`. It is also supported to use aggregation options such as `HAVING` and `GROUP BY`.

```
SELECT * FROM employees PARTITION (p0, p2)
    WHERE lname LIKE 'S%';
```

```
+----|-------|-------|----------|---------------+
| id | fname | lname | store_id | department_id |
+----|-------|-------|----------|---------------+
|  4 | Jim  | Smith |       2 |             4 |
| 11 | Jill | Stone |       1 |             4 |
+----|-------|-------|----------|---------------+
2 rows in set (0.00 sec)
```

```
SELECT id, CONCAT(fname, ' ', lname) AS name
    FROM employees PARTITION (p0) ORDER BY lname;
```

```
+----|----------------+
| id | name           |
+----|----------------+
|  3 | Ellen Johnson  |
|  4 | Jim Smith      |
|  1 | Bob Taylor     |
|  2 | Frank Williams |
+----|----------------+
4 rows in set (0.06 sec)
```

```
SELECT store_id, COUNT(department_id) AS c
    FROM employees PARTITION (p1,p2,p3)
    GROUP BY store_id HAVING c > 4;
```

```
+---|----------+
| c | store_id |
+---|----------+
| 5 |        2 |
| 5 |        3 |
+---|----------+
2 rows in set (0.00 sec)
```

Partition selection is supported for all types of table partitioning, including range partitioning and hash partitioning. For hash partitions, if partition names are not specified, `p0`, `p1`, `p2`,..., or `pN-1` is automatically used as the partition name.

`SELECT` in `INSERT ... SELECT` can also use partition selection.

4.1.8.5 Restrictions and limitations on partitions

This section introduces some restrictions and limitations on partitioned tables in TiDB.

4.1.8.5.1 Partitioning keys, primary keys and unique keys

This section discusses the relationship of partitioning keys with primary keys and unique keys. The rule governing this relationship can be expressed as follows: **Every unique key on the table must use every column in the table's partitioning expression**. This also includes the table's primary key, because it is by definition a unique key.

For example, the following table creation statements are invalid:

```
CREATE TABLE t1 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
```

```
    UNIQUE KEY (col1, col2)
)

PARTITION BY HASH(col3)
PARTITIONS 4;

CREATE TABLE t2 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    UNIQUE KEY (col1),
    UNIQUE KEY (col3)
)

PARTITION BY HASH(col1 + col3)
PARTITIONS 4;
```

In each case, the proposed table has at least one unique key that does not include all columns used in the partitioning expression.

The valid statements are as follows:

```
CREATE TABLE t1 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    UNIQUE KEY (col1, col2, col3)
)

PARTITION BY HASH(col3)
PARTITIONS 4;

CREATE TABLE t2 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    UNIQUE KEY (col1, col3)
)

PARTITION BY HASH(col1 + col3)
PARTITIONS 4;
```

The following example displays an error:

```
CREATE TABLE t3 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    UNIQUE KEY (col1, col2),
    UNIQUE KEY (col3)
)

PARTITION BY HASH(col1 + col3)
PARTITIONS 4;
```

```
ERROR 1491 (HY000): A PRIMARY KEY must include all columns in the table's
    ↪ partitioning function
```

The `CREATE TABLE` statement fails because both `col1` and `col3` are included in the proposed partitioning key, but neither of these columns is part of both of unique keys on the table. After the following modifications, the `CREATE TABLE` statement becomes valid:

```
CREATE TABLE t3 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    UNIQUE KEY (col1, col2, col3),
    UNIQUE KEY (col1, col3)
)
PARTITION BY HASH(col1 + col3)
    PARTITIONS 4;
```

The following table cannot be partitioned at all, because there is no way to include in a partitioning key any columns that belong to both unique keys:

```
CREATE TABLE t4 (
    col1 INT NOT NULL,
    col2 INT NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    UNIQUE KEY (col1, col3),
    UNIQUE KEY (col2, col4)
);
```

Because every primary key is by definition a unique key, so the next two statements are invalid:

```
CREATE TABLE t5 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    PRIMARY KEY(col1, col2)
)

PARTITION BY HASH(col3)
PARTITIONS 4;

CREATE TABLE t6 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    PRIMARY KEY(col1, col3),
    UNIQUE KEY(col2)
)

PARTITION BY HASH( YEAR(col2) )
PARTITIONS 4;
```

In the above examples, the primary key does not include all columns referenced in the partitioning expression. After adding the missing column in the primary key, the `CREATE TABLE` statement becomes valid:

```
CREATE TABLE t5 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    PRIMARY KEY(col1, col2, col3)
)
PARTITION BY HASH(col3)
PARTITIONS 4;
CREATE TABLE t6 (
    col1 INT NOT NULL,
    col2 DATE NOT NULL,
    col3 INT NOT NULL,
    col4 INT NOT NULL,
    PRIMARY KEY(col1, col2, col3),
    UNIQUE KEY(col2)
)
PARTITION BY HASH( YEAR(col2) )
```

```
PARTITIONS 4;
```

If a table has neither unique keys nor primary keys, then this restriction does not apply.

When you change tables using DDL statements, you also need to consider this restriction when adding a unique index. For example, when you create a partitioned table as shown below:

```
CREATE TABLE t_no_pk (c1 INT, c2 INT)
    PARTITION BY RANGE(c1) (
        PARTITION p0 VALUES LESS THAN (10),
        PARTITION p1 VALUES LESS THAN (20),
        PARTITION p2 VALUES LESS THAN (30),
        PARTITION p3 VALUES LESS THAN (40)
    );
```

```
Query OK, 0 rows affected (0.12 sec)
```

You can add a non-unique index by using `ALTER TABLE` statements. But if you want to add a unique index, the `c1` column must be included in the unique index.

4.1.8.5.2 Partitioning limitations relating to functions

Only the functions shown in the following list are allowed in partitioning expressions:

```
ABS()
CEILING() (see CEILING() and FLOOR())
DATEDIFF()
DAY()
DAYOFMONTH()
DAYOFWEEK()
DAYOFYEAR()
EXTRACT() (see EXTRACT() function with WEEK specifier)
FLOOR() (see CEILING() and FLOOR())
HOUR()
MICROSECOND()
MINUTE()
MOD()
MONTH()
QUARTER()
SECOND()
TIME_TO_SEC()
TO_DAYS()
TO_SECONDS()
UNIX_TIMESTAMP() (with TIMESTAMP columns)
WEEKDAY()
YEAR()
```

```
YEARWEEK()
```

4.1.8.5.3 Compatibility with MySQL

Currently, TiDB only supports range partitioning and hash partitioning. Other partitioning types that are available in MySQL such as list partitioning and key partitioning are not supported yet in TiDB.

For a table partitioned by `RANGE COLUMNS`, currently TiDB only supports using a single partitioning column.

With regard to partition management, any operation that requires moving data in the bottom implementation is not supported currently, including but not limited to: adjust the number of partitions in a hash partitioned table, modify the range of a range partitioned table, merge partitions and exchange partitions.

For the unsupported partitioning types, when you create a table in TiDB, the partitioning information is ignored and the table is created in the regular form with a warning reported. `INFORMATION_SCHEMA.PARTITION` tables are not supported currently in TiDB.

The `LOAD DATA` syntax does not support partition selection currently in TiDB.

```
create table t (id int, val int) partition by hash(id) partitions 4;
```

The regular `LOAD DATA` operation is supported:

```
load local data infile "xxx" into t ...
```

But `Load Data` does not support partition selection:

```
load local data infile "xxx" into t partition (p1)...
```

For a partitioned table, the result returned by `select * from t` is unordered between the partitions. This is different from the result in MySQL, which is ordered between the partitions but unordered inside the partitions.

```
create table t (id int, val int) partition by range (id) (
    partition p0 values less than (3),
    partition p1 values less than (7),
    partition p2 values less than (11));
```

```
Query OK, 0 rows affected (0.10 sec)
```

```
insert into t values (1, 2), (3, 4),(5, 6),(7,8),(9,10);
```

```
Query OK, 5 rows affected (0.01 sec)
Records: 5 Duplicates: 0 Warnings: 0
```

TiDB returns a different result every time, for example:

```
select * from t;
```

```
+------|------+
| id   | val |
+------|------+
|    7 |    8 |
|    9 |   10 |
|    1 |    2 |
|    3 |    4 |
|    5 |    6 |
+------|------+
5 rows in set (0.00 sec)
```

The result returned in MySQL:

```
select * from t;
```

```
+------|------+
| id   | val |
+------|------+
|    1 |    2 |
|    3 |    4 |
|    5 |    6 |
|    7 |    8 |
|    9 |   10 |
+------|------+
5 rows in set (0.00 sec)
```

4.1.9 Character Set Support

A character set is a set of symbols and encodings. A collation is a set of rules for comparing characters in a character set.

Currently, TiDB supports the following character sets:

```
mysql> SHOW CHARACTER SET;
+---------/---------------/-------------------/--------+
| Charset | Description | Default collation | Maxlen |
+---------/---------------/-------------------/--------+
| utf8   | UTF-8 Unicode | utf8_bin      |      3 |
| utf8mb4 | UTF-8 Unicode | utf8mb4_bin |      4 |
| ascii  | US ASCII     | ascii_bin       |      1 |
| latin1 | Latin1       | latin1_bin      |      1 |
| binary | binary       | binary          |      1 |
+---------/---------------/-------------------/--------+
```

```
5 rows in set (0.00 sec)
```

Warning:

TiDB incorrectly treats latin1 as a subset of utf8. This can lead to unexpected behaviors when you store characters that differ between latin1 and utf8 encodings. It is strongly recommended to the utf8mb4 character set. See TiDB #18955 for more details.

Note:

- In `TiDB`, `utf8` is treated as `utf8mb4`.
- Each character set corresponds to only one default collation.

4.1.9.1 Collation support

TiDB only supports binary collations. This means that unlike MySQL, in TiDB string comparisons are both case sensitive and accent sensitive:

```
mysql> SELECT * FROM information_schema.collations;
+--
   ↪ -------------+--------------------+------+-----------+-------------+---------+
   ↪
| COLLATION_NAME | CHARACTER_SET_NAME | ID | IS_DEFAULT | IS_COMPILED |
   ↪ SORTLEN |
+--
   ↪ -------------+--------------------+------+-----------+-------------+---------+
   ↪
| utf8mb4_bin  | utf8mb4         |   46 | Yes       | Yes         |       1 |
| latin1_bin   | latin1          |   47 | Yes       | Yes         |       1 |
| binary       | binary          |   63 | Yes       | Yes         |       1 |
| ascii_bin    | ascii           |   65 | Yes       | Yes         |       1 |
| utf8_bin     | utf8            |   83 | Yes       | Yes         |       1 |
+--
   ↪ -------------+--------------------+------+-----------+-------------+---------+
   ↪
5 rows in set (0.00 sec)

mysql> SHOW COLLATION WHERE Charset = 'utf8mb4';
```

```
+-------------+---------+------+---------+----------+---------+
| Collation | Charset | Id | Default | Compiled | Sortlen |
+-------------+---------+------+---------+----------+---------+
| utf8mb4_bin | utf8mb4 | 46 | Yes  | Yes     |      1 |
+-------------+---------+------+---------+----------+---------+
1 row in set (0.00 sec)
```

For compatibility with MySQL, TiDB will allow other collation names to be used:

```
mysql> CREATE TABLE t1 (a INT NOT NULL PRIMARY KEY AUTO_INCREMENT, b
    ↪ VARCHAR(10)) COLLATE utf8mb4_unicode_520_ci;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO t1 VALUES (1, 'a');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM t1 WHERE b = 'a';
+---+------+
| a | b   |
+---+------+
| 1 | a   |
+---+------+
1 row in set (0.00 sec)

mysql> SELECT * FROM t1 WHERE b = 'A';
Empty set (0.00 sec)

mysql> SHOW CREATE TABLE t1\G
*************************** 1. row ***************************
       Table: t1
Create Table: CREATE TABLE `t1` (
  `a` int(11) NOT NULL AUTO_INCREMENT,
  `b` varchar(10) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin DEFAULT NULL,
  PRIMARY KEY (`a`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_520_ci
    ↪ AUTO_INCREMENT=30002
1 row in set (0.00 sec)
```

4.1.9.2 Cluster character set and collation

Not supported yet.

4.1.9.3 Database character set and collation

Each database has a character set and a collation. You can use the following statements to specify the database character set and collation:

```
CREATE DATABASE db_name
    [[DEFAULT] CHARACTER SET charset_name]
    [[DEFAULT] COLLATE collation_name]

ALTER DATABASE db_name
    [[DEFAULT] CHARACTER SET charset_name]
    [[DEFAULT] COLLATE collation_name]
```

`DATABASE` can be replaced with `SCHEMA` here.

Different databases can use different character sets and collations. Use the `character_set_database` and `collation_database` to see the character set and collation of the current database:

```
mysql> CREATE SCHEMA test1 CHARACTER SET utf8 COLLATE utf8_general_ci;
Query OK, 0 rows affected (0.09 sec)

mysql> USE test1;
Database changed
mysql> SELECT @@character_set_database, @@collation_database;
+--------------------------/----------------------+
| @@character_set_database | @@collation_database |
+--------------------------/----------------------+
| utf8                     | utf8_general_ci      |
+--------------------------/----------------------+
1 row in set (0.00 sec)

mysql> CREATE SCHEMA test2 CHARACTER SET latin1 COLLATE latin1_general_ci;
Query OK, 0 rows affected (0.09 sec)

mysql> use test2;
Database changed
mysql> SELECT @@character_set_database, @@collation_database;
+--------------------------/----------------------+
| @@character_set_database | @@collation_database |
+--------------------------/----------------------+
| latin1                   | latin1_general_ci    |
+--------------------------/----------------------+
1 row in set (0.00 sec)
```

You can also see the two values in INFORMATION_SCHEMA:

```
SELECT DEFAULT_CHARACTER_SET_NAME, DEFAULT_COLLATION_NAME
FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = 'db_name';
```

4.1.9.4 Table character set and collation

You can use the following statement to specify the character set and collation for tables:

```
CREATE TABLE tbl_name (column_list)
    [[DEFAULT] CHARACTER SET charset_name]
    [COLLATE collation_name]]

ALTER TABLE tbl_name
    [[DEFAULT] CHARACTER SET charset_name]
    [COLLATE collation_name]
```

For example:

```
mysql> CREATE TABLE t1(a int) CHARACTER SET utf8 COLLATE utf8_general_ci;
Query OK, 0 rows affected (0.08 sec)
```

The database character set and collation are used as the default values for table definitions if the table character set and collation are not specified in individual column definitions.

4.1.9.5 Column character set and collation

See the following table for the character set and collation syntax for columns:

```
col_name {CHAR | VARCHAR | TEXT} (col_length)
    [CHARACTER SET charset_name]
    [COLLATE collation_name]

col_name {ENUM | SET} (val_list)
    [CHARACTER SET charset_name]
    [COLLATE collation_name]
```

The table character set and collation are used as the default values for column definitions if the column character set and collation are not specified in individual column definitions.

4.1.9.6 String character sets and collation

Each character literal in a string has a character set and a collation. When you use a string, this option is available:

```
[_charset_name]'string' [COLLATE collation_name]
```

Example:

```
SELECT 'string';
SELECT _latin1'string';
SELECT _latin1'string' COLLATE latin1_danish_ci;
```

Rules:

- Rule 1: If you specify `CHARACTER SET charset_name` and `COLLATE collation_name`, then `CHARACTER SET charset_name` and `COLLATE collation_name` are used directly.
- Rule 2: If you specify `CHARACTER SET charset_name` but do not specify `COLLATE collation_name`, `CHARACTER SET charset_name` and the default collation of `CHARACTER SET charset_name` are used.
- Rule 3: If you specify neither `CHARACTER SET charset_name` nor `COLLATE collation_name`, the character set and collation given by the system variables `character_set_connection` and `collation_connection` are used.

4.1.9.7 Connection character sets and collations

- The server character set and collation are the values of the `character_set_server` and `collation_server` system variables.
- The character set and collation of the default database are the values of the `character_set_database` and `collation_database` system variables. You can use `character_set_connection` and `collation_connection` to specify the character set and collation for each connection. The `character_set_client` variable is to set the client character set. Before returning the result, the `character_set_results` system variable indicates the character set in which the server returns query results to the client, including the metadata of the result.

You can use the following statement to specify a particular collation that is related to the client:

- `SET NAMES 'charset_name' [COLLATE 'collation_name']`

  `SET NAMES` indicates what character set the client will use to send SQL statements to the server. `SET NAMES utf8` indicates that all the requests from the client use utf8, as well as the results from the server.

  The `SET NAMES 'charset_name'` statement is equivalent to the following statement combination:

```
SET character_set_client = charset_name;
SET character_set_results = charset_name;
SET character_set_connection = charset_name;
```

  `COLLATE` is optional, if absent, the default collation of the `charset_name` is used.

- `SET CHARACTER SET 'charset_name'`

  Similar to `SET NAMES`, the `SET NAMES 'charset_name'` statement is equivalent to the following statement combination:

```
SET character_set_client = charset_name;
SET character_set_results = charset_name;
SET collation_connection = @@collation_database;
```

4.1.9.8 Optimization levels of character sets and collations

String => Column => Table => Database => Server => Cluster

4.1.9.9 General rules on selecting character sets and collation

- Rule 1: If you specify `CHARACTER SET charset_name` and `COLLATE collation_name`, then `CHARACTER SET charset_name` and `COLLATE collation_name` are used directly.
- Rule 2: If you specify `CHARACTER SET charset_name` and do not specify `COLLATE collation_name`, then `CHARACTER SET charset_name` and the default comparison collation of `CHARACTER SET charset_name` are used.
- Rule 3: If you specify neither `CHARACTER SET charset_name` nor `COLLATE collation_name`, the character set and collation with higher optimization levels are used.

4.1.9.10 Validity check of characters

For the specified `utf8` or `utf8mb4` character set, TiDB only supports the valid `utf8` character, and reports the `incorrect utf8 value` error when the character is invalid. This validity check of characters in TiDB is compatible with MySQL 8.0 but incompatible with MySQL 5.7 or earlier versions.

To disable this error reporting, use `set @@tidb_skip_utf8_check=1;` to skip the character check.

For more information, see Connection Character Sets and Collations in MySQL.

— title: SQL Mode summary: Learn SQL mode. aliases: [‘/docs/v3.0/sql-mode/’,‘/docs/v3.0/reference/sql/sql-mode/’] —

4.1.10 SQL Mode

TiDB servers operate in different SQL modes and apply these modes differently for different clients. SQL mode defines the SQL syntaxes that TiDB supports and the type of data validation check to perform, as described below:

After TiDB is started, modify `SET [ SESSION | GLOBAL ] sql_mode='modes'` to set SQL mode.

Ensure that you have `SUPER` privilege when setting SQL mode at `GLOBAL` level, and your setting at this level only affects the connections established afterwards. Changes to SQL mode at `SESSION` level only affect the current client.

`Modes` are a series of different modes separated by commas (‘,’). You can use the `SELECT @@sql_mode` statement to check the current SQL mode. The default value of SQL mode: `ONLY_FULL_GROUP_BY, STRICT_TRANS_TABLES, NO_ZERO_IN_DATE, NO_ZERO_DATE, ERROR_FOR_DIVISION_BY_ZERO, NO_AUTO_CREATE_USER, NO_ENGINE_SUBSTITUTION`.

4.1.10.1 Important `sql_mode` values

- `ANSI`: This mode complies with standard SQL. In this mode, data is checked. If data does not comply with the defined type or length, the data type is adjusted or trimmed and a `warning` is returned.
- `STRICT_TRANS_TABLES`: Strict mode, where data is strictly checked. When any incorrect data is inserted into a table, an error is returned.
- `TRADITIONAL`: In this mode, TiDB behaves like a "traditional" SQL database system. An error instead of a warning is returned when any incorrect value is inserted into a column. Then, the `INSERT` or `UPDATE` statement is immediately stopped.

4.1.10.2 SQL mode table

Name	Description
`PIPES_AS_CONCAT`	Treat `||` as a string concatenation operator (`+`) (the same as `CONCAT()`), not as an `OR` (full support)

Name	Description
ANSI_QUOTES	Treats " as an identifier. If ANSI_QUOTES is enabled, only single quotes are treated as string literals, and double quotes are treated as identifiers. Therefore, double quotes cannot be used to quote strings. (full support)

Name	Description
IGNORE_SPACE	If this mode is enabled, the system ignores space. For example: “user” and “user” are the same. (full support)

Name	Description
ONLY_FULL_GROUP_BY	If a non-aggregated column that is referred to in SELECT, HAVING, or ORDER BY is absent in GROUP BY, this SQL statement is invalid, because it is abnormal for a column to be absent in GROUP BY

Name	Description
NO_UNSIGNED_SUBTRACTION ↪	Do not mark the result as UNSIGNED ↪ if an operand has no symbol in subtraction. (full support)

Name	Description
NO_DIR_IN_CREATE	Ignore all INDEX DIRECTORY and DATA DIRECTORY directives when a table is created. This option is only useful for secondary replication servers (syntax support only)

Name	Description
NO_KEY_OPTIONS	When you use the SHOW CREATE TABLE statement, MySQL-specific syntaxes such as ENGINE are not exported. Consider this option when migrating across DB types using mysqldump. (syntax support only)

Name	Description
NO_FIELD_OPTIONS	Value you use the SHOW CREATE TABLE statement, MySQL-specific syntaxes such as ENGINE are not exported. Consider this option when migrating across DB types using mysqldump. (syntax support only)

Name	Description
NO_TABLE_OPTIONS	When you use the SHOW CREATE TABLE statement, MySQL-specific syntaxes such as ENGINE are not exported. Consider this option when migrating across DB types using mysqldump. (syntax support only)

Name	Description
NO_AUTO_VALUE_ON_ZERO	If this mode is enabled, when the value passed in the AUTO_INCREMENT column is 0 or a specific value, the system directly writes this value to this column. When NULL is passed, the system automatically generates the next serial

Name	Description
NO_BACKSLASH_ESCAPES ↪	In this mode is enabled, the \ backslash symbol only stands for itself. (full support)

Name	Description
STRICT_TRANS_TABLES	Treats the strict mode for the transaction storage engine and rolls back the entire statement after an illegal value is inserted. (full support)

Name	Description
STRICT_ALL_TABLES	All transactional tables, rolls back the entire transaction statement after an illegal value is inserted. (full support)

Name	Description
NO_ZERO_IN_DATE	Strict mode, where dates with a month or day part of 0 are not accepted. If you use the IGNORE option, TiDB inserts '0000-00-00' for a similar date. In non-strict mode, this date is accepted but a warning is returned. (full sup-

Name	Description
NO_ZERO_DATE	Does not use '0000-00-00' as a legal date in strict mode. You can still insert a zero date with the IGNORE option. In non-strict mode, this date is accepted but a warning is returned. (full support)

Name	Description
`ALLOW_INVALID_DATES`	In this mode, the system does not check the validity of all dates. It only checks the month value ranging from `1` to `12` and the date value ranging from `1` to `31`. The mode only applies to `DATE` and `DATATIME` columns. All `TIMESTAMP`

Name	Description
ERROR_FOR_DIVISION_BY_ZERO	If this mode is enabled, the system returns an error when handling division by 0 in data-change operations (INSERT or UPDATE). If this mode is not enabled, the system returns a warning and NULL is used

Name	Description
NO_AUTO_CREATE_USER	Prevent GRANT from automatically creating new users, except for the specified password (full support)

Name	Description
HIGH_NOT_PRECEDENCE	The precedence of the NOT operator is such that expressions such as NOT a BETWEEN b AND c are parsed as NOT (a BETWEEN b AND c). In some older versions of MySQL,

Name	Description
NO_ENGINE_SUBSTITUTION	Prevents the automatic replacement of storage engines if the required storage engine is disabled or not compiled. (syntax support only)

Name	Description
PAD_CHAR_TO_FULL_LENGTH ↪	If this mode is enabled, the system does not trim the trailing spaces for CHAR types. (syntax support only. This mode has been deprecated in MySQL 8.0.)

Name	Description
REAL_AS_FLOAT	Treat REAL as the synonym of FLOAT , not the synonym of DOUBLE (full support)
POSTGRESQL	Equivalent to PIPES_AS_CONCAT , ANSI_QUOTES , IGNORE_SPACE , NO_KEY_OPTIONS , NO_TABLE_OPTIONS , NO_FIELD_OPTIONS (syntax support only)

Name	Description
MSSQL	Equivalent to PIPES_AS_CONCAT , ANSI_QUOTES , IGNORE_SPACE , NO_KEY_OPTIONS , NO_TABLE_OPTIONS , NO_FIELD_OPTIONS (syntax support only)
DB2	Equivalent to PIPES_AS_CONCAT , ANSI_QUOTES , IGNORE_SPACE , NO_KEY_OPTIONS , NO_TABLE_OPTIONS , NO_FIELD_OPTIONS (syntax support only)

Name	Description
MAXDB	Equivalent to PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS, NO_AUTO_CREATE_USER (full support)
MySQL323	Equivalent to NO_FIELD_OPTIONS, HIGH_NOT_PRECEDENCE (syntax support only)
MYSQL40	Equivalent to NO_FIELD_OPTIONS, HIGH_NOT_PRECEDENCE (syntax support only)

Name	Description
`ANSI`	Equivalent to `REAL_AS_FLOAT` ↪ , `PIPES_AS_CONCAT` ↪ , `ANSI_QUOTES` ↪ , `IGNORE_SPACE` ↪ (syntax support only)
`TRADITIONAL` ↪	Equivalent to `STRICT_TRANS_TABLES` ↪ , `STRICT_ALL_TABLES` ↪ , `NO_ZERO_IN_DATE` ↪ , `NO_ZERO_DATE` ↪ , `ERROR_FOR_DIVISION_BY_ZERO` ↪ , `NO_AUTO_CREATE_USER` ↪ (syntax support only)

Name	Description
ORACLE	Equivalent to PIPES_AS_CONCAT, ANSI_QUOTES, IGNORE_SPACE, NO_KEY_OPTIONS, NO_TABLE_OPTIONS, NO_FIELD_OPTIONS, NO_AUTO_CREATE_USER (syntax support only)

4.1.11 Views

TiDB supports views. A view acts as a virtual table, whose schema is defined by the `SELECT` statement that creates the view. Using views has the following benefits:

- Exposing only safe fields and data to users to ensure security of sensitive fields and data stored in the underlying table.
- Defining complex queries that frequently appear as views to make complex queries easier and more convenient.

4.1.11.1 Query views

Querying a view is similar to querying an ordinary table. However, when TiDB queries a view, it actually queries the `SELECT` statement associated with the view.

4.1.11.2 Show metadata

To obtain the metadata of views, choose any of the following methods.

4.1.11.2.1 Use the SHOW CREATE TABLE view_name or SHOW CREATE VIEW view_name statement

Usage example:

```
show create view v;
```

This statement shows the CREATE VIEW statement corresponding to this view and the value of the character_set_client and collation_connection system variables when the view was created.

```
+--
    ↪ ----+----------------------------------------------------------------------
    ↪
| View | Create View
    ↪
    ↪ | character_set_client | collation_connection |
+--
    ↪ ----+----------------------------------------------------------------------
    ↪
| v    | CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`127.0.0.1` SQL SECURITY
    ↪ DEFINER VIEW `v` (`a`) AS SELECT `s`.`a` FROM `test`.`t` LEFT JOIN `
    ↪ test`.`s` ON `t`.`a`=`s`.`a` | utf8 | utf8_general_ci |
+--
    ↪ ----+----------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)
```

4.1.11.2.2 Query the INFORMATION_SCHEMA.VIEWS table

Usage example:

```
select * from information_schema.views;
```

You can view the relevant meta information of the view by querying this table, such as TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, VIEW_DEFINITION, CHECK_OPTION, IS_UPDATABLE, DEFINER, SECURITY_TYPE, CHARACTER_SET_CLIENT, and COLLATION_CONNECTION ↪ .

```
+--
    ↪ -------------+--------------+------------+-----------------------------------
    ↪
| TABLE_CATALOG | TABLE_SCHEMA | TABLE_NAME | VIEW_DEFINITION
    ↪                                          | CHECK_OPTION | IS_UPDATABLE
    ↪  | DEFINER   | SECURITY_TYPE | CHARACTER_SET_CLIENT |
    ↪ COLLATION_CONNECTION |
```

```
+--
    ↪ ------------+-------------+-----------+-----------------------------------
    ↪
| def          | test        | v         | SELECT `s`.`a` FROM `test`.`t` LEFT
    ↪  JOIN `test`.`s` ON `t`.`a`=`s`.`a` | CASCADED | NO  | root@127.0.0.1
    ↪ | DEFINER    | utf8                | utf8_general_ci    |
+--
    ↪ ------------+-------------+-----------+-----------------------------------
    ↪
1 row in set (0.00 sec)
```

4.1.11.2.3 Use the HTTP APIs

Usage example:

```
curl http://127.0.0.1:10080/schema/test/v
```

By visiting `http://{TiDBIP}:10080/schema/{db}/{view}`, you can get all the metadata for the view.

```
{
 "id": 122,
 "name": {
  "O": "v",
  "L": "v"
 },
 "charset": "utf8",
 "collate": "utf8_general_ci",
 "cols": [
  {
   "id": 1,
   "name": {
    "O": "a",
    "L": "a"
   },
   "offset": 0,
   "origin_default": null,
   "default": null,
   "default_bit": null,
   "default_is_expr": false,
   "generated_expr_string": "",
   "generated_stored": false,
   "dependences": null,
   "type": {
    "Tp": 0,
```

```
   "Flag": 0,
   "Flen": 0,
   "Decimal": 0,
   "Charset": "",
   "Collate": "",
   "Elems": null
  },
  "state": 5,
  "comment": "",
  "hidden": false,
  "version": 0
 }
],
"index_info": null,
"fk_info": null,
"state": 5,
"pk_is_handle": false,
"is_common_handle": false,
"comment": "",
"auto_inc_id": 0,
"auto_id_cache": 0,
"auto_rand_id": 0,
"max_col_id": 1,
"max_idx_id": 0,
"update_timestamp": 416801600091455490,
"ShardRowIDBits": 0,
"max_shard_row_id_bits": 0,
"auto_random_bits": 0,
"pre_split_regions": 0,
"partition": null,
"compression": "",
"view": {
 "view_algorithm": 0,
 "view_definer": {
  "Username": "root",
  "Hostname": "127.0.0.1",
  "CurrentUser": false,
  "AuthUsername": "root",
  "AuthHostname": "%"
 },
 "view_security": 0,
 "view_select": "SELECT `s`.`a` FROM `test`.`t` LEFT JOIN `test`.`s` ON `t
    ↪ `.`a`=`s`.`a`",
 "view_checkoption": 1,
 "view_cols": null
```

```
 },
 "sequence": null,
 "Lock": null,
 "version": 3,
 "tiflash_replica": null
}
```

4.1.11.3 Example

The following example creates a view, queries this view, and delete this view:

```
create table t(a int, b int);
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
insert into t values(1, 1),(2,2),(3,3);
```

```
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

```
create table s(a int);
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
insert into s values(2),(3);
```

```
Query OK, 2 rows affected (0.01 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

```
create view v as select s.a from t left join s on t.a = s.a;
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
select * from v;
```

```
+------+
| a    |
+------+
| NULL |
|    2 |
|    3 |
+------+
3 rows in set (0.00 sec)
```

```
drop view v;
```

```
Query OK, 0 rows affected (0.02 sec)
```

4.1.11.4 Limitations

Currently, views in TiDB are subject to the following limitations:

- Materialized views are not supported yet.
- Views in TiDB are read-only and do not support write operations such as `UPDATE`, `INSERT`, `DELETE`, and `TRUNCATE`.
- For created views, the only supported DDL operation is `DROP [VIEW | TABLE]`

4.1.11.5 See also

- CREATE VIEW
- DROP VIEW

4.2 Configuration

4.2.1 tidb-server

4.2.1.1 The System Variables

The system variables in MySQL are the system parameters that modify the operation of the database runtime. These variables have two types of scope, Global Scope and Session Scope. TiDB supports all the system variables in MySQL 5.7. Most of the variables are only supported for compatibility and do not affect the runtime behaviors.

4.2.1.1.1 Set the system variables

You can use the SET statement to change the value of the system variables. Before you change, consider the scope of the variable. For more information, see MySQL Dynamic System Variables.

Set Global variables

Add the `GLOBAL` keyword before the variable or use `@@global.` as the modifier:

```
SET GLOBAL autocommit = 1;
SET @@global.autocommit = 1;
```

> **Note:**
>
> In a distributed TiDB database, a variable's `GLOBAL` setting is persisted to the storage layer. A single TiDB instance proactively gets the `GLOBAL` information and forms `gvc` (global variables cache) every 2 seconds. The cache information remains valid within 2 seconds. When you set the `GLOBAL` variable, to ensure the effectiveness of new sessions, make sure that the interval between two operations is larger than 2 seconds. For details, see Issue #14531.

Set Session variables

Add the `SESSION` keyword before the variable, use `@@session.` as the modifier, or use no modifier:

```
SET SESSION autocommit = 1;
SET @@session.autocommit = 1;
SET @@autocommit = 1;
```

> **Note:**
>
> `LOCAL` and `@@local.` are the synonyms for `SESSION` and `@@session.`

The working mechanism of system variables

- Session variables will only initialize their own values based on global variables when a session is created. Changing a global variable does not change the value of the system variable being used by the session that has already been created.

```
mysql> SELECT @@GLOBAL.autocommit;
+--------------------+
| @@GLOBAL.autocommit |
+--------------------+
| ON                 |
+--------------------+
1 row in set (0.00 sec)

mysql> SELECT @@SESSION.autocommit;
+---------------------+
| @@SESSION.autocommit |
+---------------------+
```

```
| ON                  |
+---------------------+
1 row in set (0.00 sec)

mysql> SET GLOBAL autocommit = OFF;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT @@SESSION.autocommit; -- Session variables do not change
    ↪ , and the transactions in the session are executed in the form
    ↪ of autocommit.
+----------------------+
| @@SESSION.autocommit |
+----------------------+
| ON                   |
+----------------------+
1 row in set (0.00 sec)

mysql> SELECT @@GLOBAL.autocommit;
+----------------------+
| @@GLOBAL.autocommit |
+----------------------+
| OFF                  |
+----------------------+
1 row in set (0.00 sec)

mysql> exit
Bye
$ mysql -h127.0.0.1 -P4000 -uroot -D test
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.25-TiDB-None MySQL Community Server (Apache License
    ↪  2.0)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights
    ↪ reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
    ↪ statement.

mysql> SELECT @@SESSION.autocommit; -- The newly created session uses
    ↪ a new global variable.
```

```
+----------------------+
| @@SESSION.autocommit |
+----------------------+
| OFF                  |
+----------------------+
1 row in set (0.00 sec)
```

4.2.1.1.2 The fully supported MySQL system variables in TiDB

The following MySQL system variables are fully supported in TiDB and have the same behaviors as in MySQL.

Name	Scope	Description
autocommit	GLOBAL | SESSION	whether automatically commit a transaction
sql_mode	GLOBAL | SESSION	support some of the MySQL SQL modes
time_zone	GLOBAL | SESSION	the time zone of the database
tx_isolation	GLOBAL | SESSION	the isolation level of a transaction
max_execution_time	GLOBAL | SESSION	the execution timeout for a statement, in milliseconds
innodb_lock_wait_timeout	GLOBAL | SESSION	the lock wait time for pessimistic transactions, in seconds
interactive_timeout	SESSION | GLOBAL	the idle timeout of the interactive user session, in seconds

Note:

Unlike in MySQL, the `max_execution_time` system variable currently works on all kinds of statements in TiDB, not only restricted to the `SELECT` statement. The precision of the timeout value is roughly 100ms. This means the statement might not be terminated in accurate milliseconds as you specify.

4.2.1.1.3 TiDB specific system variables

See TiDB Specific System Variables.

4.2.1.2 TiDB Specific System Variables

TiDB contains a number of system variables which are specific to its usage, and **do not** apply to MySQL. These variables start with a `tidb_` prefix, and can be tuned to optimize system performance.

4.2.1.2.1 System variables

Variables can be set with the `SET` statement, for example:

```
set @@tidb_distsql_scan_concurrency = 10
```

If you need to set the global variable, run:

```
set @@global.tidb_distsql_scan_concurrency = 10
```

tidb_snapshot

- Scope: SESSION
- Default value: “”
- This variable is used to set the time point at which the data is read by the session. For example, when you set the variable to “2017-11-11 20:20:20” or a TSO number like “400036290571534337”, the current session reads the data of this moment.

tidb_import_data

- Scope: SESSION
- Default value: 0
- This variable indicates whether to import data from the dump file currently.
- To speed up importing, the unique index constraint is not checked when the variable is set to 1.
- This variable is only used by TiDB Lightning. Do not modify it.

tidb_opt_agg_push_down

- Scope: SESSION
- Default value: 0
- This variable is used to set whether the optimizer executes the optimization operation of pushing down the aggregate function to the position before Join.
- When the aggregate operation is slow in query, you can set the variable value to 1.

tidb_auto_analyze_ratio

- Scope: GLOBAL
- Default value: 0.5
- This variable is used to set the threshold when TiDB automatically executes `ANALYZE TABLE` in a background thread to update table statistics. For example, a value of 0.5 means that auto-analyze is triggered when greater than 50% of the rows in a table have been modified. Auto-analyze can be restricted to only execute during certain hours of the day by specifying `tidb_auto_analyze_start_time` and `tidb_auto_analyze_end_time`.

> **Note:**
>
> Only when the `run-auto-analyze` option is enabled in the starting configuration file of TiDB, the `auto_analyze` feature can be triggered.

tidb_auto_analyze_start_time

- Scope: GLOBAL
- Default value: 00:00 +0000
- This variable is used to restrict the time window that the automatic update of statistics is permitted. For example, to only allow automatic statistics updates between 1AM and 3AM, set `tidb_auto_analyze_start_time='01:00 +0000'` and `tidb_auto_analyze_end_time='03:00 +0000'`.

tidb_auto_analyze_end_time

- Scope: GLOBAL
- Default value: 23:59 +0000
- This variable is used to restrict the time window that the automatic update of statistics is permitted. For example, to only allow automatic statistics updates between 1AM and 3AM, set `tidb_auto_analyze_start_time='01:00 +0000'` and `tidb_auto_analyze_end_time='03:00 +0000'`.

tidb_build_stats_concurrency

- Scope: SESSION
- Default value: 4
- This variable is used to set the concurrency of executing the `ANALYZE` statement.
- When the variable is set to a larger value, the execution performance of other queries is affected.

tidb_checksum_table_concurrency

- Scope: SESSION
- Default value: 4
- This variable is used to set the scan index concurrency of executing the `ADMIN CHECKSUM TABLE` statement.
- When the variable is set to a larger value, the execution performance of other queries is affected.

tidb_distsql_scan_concurrency

- Scope: SESSION | GLOBAL
- Default value: 15
- This variable is used to set the concurrency of the `scan` operation.
- Use a bigger value in OLAP scenarios, and a smaller value in OLTP scenarios.
- For OLAP scenarios, the maximum value cannot exceed the number of CPU cores of all the TiKV nodes.

tidb_current_ts

- Scope: SESSION
- Default value: 0
- This variable is read-only. It is used to obtain the timestamp of the current transaction.

tidb_config

- Scope: SESSION
- Default value: “”
- This variable is read-only. It is used to obtain the configuration information of the current TiDB server.

tidb_index_lookup_size

- Scope: SESSION | GLOBAL
- Default value: 20000
- This variable is used to set the batch size of the `index lookup` operation.
- Use a bigger value in OLAP scenarios, and a smaller value in OLTP scenarios.

tidb_index_lookup_concurrency

- Scope: SESSION | GLOBAL
- Default value: 4
- This variable is used to set the concurrency of the `index lookup` operation.
- Use a bigger value in OLAP scenarios, and a smaller value in OLTP scenarios.

tidb_index_lookup_join_concurrency

- Scope: SESSION | GLOBAL
- Default value: 4
- This variable is used to set the concurrency of the `index lookup join` algorithm.

tidb_hash_join_concurrency

- Scope: SESSION | GLOBAL
- Default value: 5
- This variable is used to set the concurrency of the `hash join` algorithm.

tidb_index_serial_scan_concurrency

- Scope: SESSION | GLOBAL
- Default value: 1
- This variable is used to set the concurrency of the `serial scan` operation.
- Use a bigger value in OLAP scenarios, and a smaller value in OLTP scenarios.

tidb_projection_concurrency

- Scope: SESSION | GLOBAL
- Default value: 4
- This variable is used to set the concurrency of the `Projection` operator.

tidb_union_concurrency

- Scope: SESSION | GLOBAL
- Default value: 4
- This variable is used to set the concurrency degree of the `union` operator.

tidb_hashagg_partial_concurrency

- Scope: SESSION | GLOBAL
- Default value: 4
- This variable is used to set the concurrency of executing the concurrent `hash aggregation` algorithm in the `partial` phase.
- When the parameter of the aggregate function is not distinct, `HashAgg` is run concurrently and respectively in two phases - the `partial` phase and the `final` phase.

tidb_hashagg_final_concurrency

- Scope: SESSION | GLOBAL
- Default value: 4
- This variable is used to set the concurrency of executing the concurrent `hash aggregation` algorithm in the `final` phase.
- When the parameter of the aggregate function is not distinct, `HashAgg` is run concurrently and respectively in two phases - the `partial` phase and the `final` phase.

tidb_index_join_batch_size

- Scope: SESSION | GLOBAL
- Default value: 25000
- This variable is used to set the batch size of the `index lookup join` operation.
- Use a bigger value in OLAP scenarios, and a smaller value in OLTP scenarios.

tidb_skip_utf8_check

- Scope: SESSION | GLOBAL
- Default value: 0
- This variable is used to set whether to skip UTF-8 validation.
- Validating UTF-8 characters affects the performance. When you are sure that the input characters are valid UTF-8 characters, you can set the variable value to 1.

tidb_init_chunk_size

- Scope: SESSION | GLOBAL
- Default value: 32
- This variable is used to set the number of rows for the initial chunk during the execution process.

tidb_max_chunk_size

- Scope: SESSION | GLOBAL
- Default value: 1024
- This variable is used to set the maximum number of rows in a chunk during the execution process.

tidb_mem_quota_query

- Scope: SESSION
- Default value: 32 GB
- This variable is used to set the threshold value of memory quota for a query.

- If the memory quota of a query during execution exceeds the threshold value, TiDB performs the operation designated by the OOMAction option in the configuration file.

tidb_general_log

- Scope: SERVER
- Default value: 0
- This variable is used to set whether to record all the SQL statements in the log.

tidb_retry_limit

- Scope: SESSION | GLOBAL
- Default value: 10
- This variable is used to set the maximum number of retries for optimistic transactions. When a transaction encounters retryable errors (such as transaction conflicts, very slow transaction commit, or table schema changes), this transaction is re-executed according to this variable. Note that setting `tidb_retry_limit` to `0` disables the automatic retry. This variable is applicable only to optimistic transactions, not to pessimistic transactions.

tidb_disable_txn_auto_retry

- Scope: SESSION | GLOBAL
- Default: on
- This variable is used to set whether to disable the automatic retry of explicit optimistic transactions. The default value of `on` means that transactions will not automatically retry in TiDB and `COMMIT` statements might return errors that need to be handled in the application layer.

  Setting the value to `off` means that TiDB will automatically retry transactions, resulting in fewer errors from `COMMIT` statements. Be careful when making this change, because it might result in lost updates.

  This variable does not affect automatically committed implicit transactions and internally executed transactions in TiDB. The maximum retry count of these transactions is determined by the value of `tidb_retry_limit`.

  For more details, see limits of retry.

  This variable only applies to optimistic transactions, not to pessimistic transactions. The number of retries for pessimistic transactions is controlled by `max_retry_count`.

tidb_backoff_weight

- Scope: SESSION | GLOBAL

- Default value: 2
- This variable is used to increase the weight of the maximum time of TiDB `backoff`, that is, the maximum retry time for sending a retry request when an internal network or other component (TiKV, PD) failure is encountered. This variable can be used to adjust the maximum retry time and the minimum value is 1.

  For example, the base timeout for TiDB to take TSO from PD is 15 seconds. When `tidb_backoff_weight = 2`, the maximum timeout for taking TSO is: *base time * 2 = 30 seconds*.

  In the case of a poor network environment, appropriately increasing the value of this variable can effectively alleviate error reporting to the application end caused by timeout. If the application end wants to receive the error information more quickly, minimize the value of this variable.

tidb_enable_table_partition

- Scope: SESSION
- Default value: "auto"
- This variable is used to set whether to enable the `TABLE PARTITION` feature.
  - `off` indicates disabling the `TABLE PARTITION` feature. In this case, the syntax that creates a partition table can be executed, but the table created is not a partitioned one.
  - `auto` indicates enabling the `TABLE PARTITION` feature. Currently, it indicates enabling range partition and hash partition.
  - `on` functions the same way as `auto` does.
- Currently, TiDB only supports range partition and hash partition.

tidb_backoff_lock_fast

- Scope: SESSION | GLOBAL
- Default value: 100
- This variable is used to set the `backoff` time when the read request meets a lock.

tidb_ddl_reorg_worker_cnt

- Scope: GLOBAL
- Default value (before 3.0.3): 16
- Default value (in 3.0.3 or later): 4
- This variable is used to set the concurrency of the DDL operation in the `re-organize` phase.

tidb_ddl_reorg_batch_size

- Scope: GLOBAL
- Default value (before 3.0.3): 1024
- Default value (in 3.0.3 or later): 256
- This variable is used to set the batch size during the `re-organize` phase of the DDL operation. For example, when TiDB executes the `ADD INDEX` operation, the index data needs to backfilled by `tidb_ddl_reorg_worker_cnt` (the number) concurrent workers. Each worker backfills the index data in batches.
  - If many updating operations such as `UPDATE` and `REPLACE` exist during the `ADD INDEX` operation, a larger batch size indicates a larger probability of transaction conflicts. In this case, you need to adjust the batch size to a smaller value. The minimum value is 32.
  - If the transaction conflict does not exist, you can set the batch size to a large value. The maximum value is 10240. This can increase the speed of the backfilling data, but the write pressure on TiKV also becomes higher.

tidb_ddl_reorg_priority

- Scope: SESSION
- Default value: `PRIORITY_LOW`
- This variable is used to set the priority of executing the `ADD INDEX` operation in the `re-organize` phase.
- You can set the value of this variable to `PRIORITY_LOW`, `PRIORITY_NORMAL` or `PRIORITY_HIGH`.

tidb_ddl_error_count_limit

- Scope: GLOBAL
- Default value: 512
- This variable is used to set the number of retries when the DDL operation fails. When the number of retries exceeds the parameter value, the wrong DDL operation is canceled.

tidb_max_delta_schema_count New in v3.0.5

- Scope: GLOBAL
- Default value: 1024
- This variable is used to set the maximum number of schema versions (the table IDs modified for corresponding versions) allowed to be cached. The value range is 100 ~ 16384.

tidb_force_priority

- Scope: SESSION

- Default value: `NO_PRIORITY`
- This variable is used to change the default priority for statements executed on a TiDB server. A use case is to ensure that a particular user that is performing OLAP queries receives lower priority than users performing OLTP queries.
- You can set the value of this variable to `NO_PRIORITY`, `LOW_PRIORITY`, `DELAYED` or `HIGH_PRIORITY`.

tidb_opt_write_row_id

- Scope: SESSION
- Default value: 0
- This variable is used to set whether to allow `insert`, `replace` and `update` statements to operate on the column `_tidb_rowid`. It is not allowed by default. This variable can be used only when importing data with TiDB tools.

4.2.1.2.2 SHARD_ROW_ID_BITS

For the tables with non-integer PK or without PK, TiDB uses an implicit auto-increment ROW ID. When a large number of `INSERT` operations occur, the data is written into a single Region, causing a write hot spot.

To mitigate the hot spot issue, you can configure `SHARD_ROW_ID_BITS`. The ROW ID is scattered and the data is written into multiple different Regions. But setting an overlarge value might lead to an excessively large number of RPC requests, which increases the CPU and network overheads.

- `SHARD_ROW_ID_BITS = 4` indicates 16 shards
- `SHARD_ROW_ID_BITS = 6` indicates 64 shards
- `SHARD_ROW_ID_BITS = 0` indicates the default 1 shard

Usage of statements:

- `CREATE TABLE`: `CREATE TABLE t (c int)SHARD_ROW_ID_BITS = 4;`
- `ALTER TABLE`: `ALTER TABLE t SHARD_ROW_ID_BITS = 4;`

4.2.1.2.3 tidb_slow_log_threshold

- Scope: SESSION
- Default value: 300ms
- This variable is used to output the threshold value of the time consumed by the slow log. When the time consumed by a query is larger than this value, this query is considered as a slow log and its log is output to the slow query log.

Usage example:

```
set tidb_slow_log_threshold = 200
```

4.2.1.2.4 tidb_query_log_max_len

- Scope: SESSION
- Default value: 2048 (bytes)
- The maximum length of the SQL statement output. When the output length of a statement is larger than the `tidb_query-log-max-len` value, the statement is truncated to output.

Usage example:

```
set tidb_query_log_max_len = 20
```

tidb_txn_mode New in v3.0.4

- Scope: SESSION (GLOBAL is supported since v3.0.4)
- Default value: "pessimistic"
- This variable is used to set the transaction mode. TiDB v3.0 supports the pessimistic transactions. Since TiDB v3.0.8, the pessimistic transaction mode is enabled by default.
- If you upgrade TiDB from v3.0.7 or earlier versions to v3.0.8 or later versions, the default transaction mode does not change. **Only the newly created clusters use the pessimistic transaction mode by default**.
- If this variable is set to "optimistic" or "", TiDB uses the optimistic transaction mode.

Since TiDB v3.0.4, the `tidb_txn_mode` variable supports the GLOBAL scope and can be used to set the global transaction mode. When you modify the global transaction mode, only the sessions created after the modification takes effect uses the new transaction mode.

tidb_constraint_check_in_place

- Scope: SESSION | GLOBAL
- Default value: 0
- This setting only applies to optimistic transactions. When this variable is set to zero, checking for duplicate values in UNIQUE indexes is deferred until the transaction commits. This helps improve performance, but might be an unexpected behavior for some applications. See Constraints for details.
  - When set to zero and using optimistic transactions:

```
tidb >create table t (i int key)
tidb >insert into t values (1);
tidb >begin optimistic;
tidb >insert into t values (1);
Query OK, 1 row affected
tidb >commit; -- Check only when a transaction is committed.
ERROR 1062 : Duplicate entry '1' for key 'PRIMARY'
```

  - When set to 1 and using optimistic transactions:

```
tidb >set @@tidb_constraint_check_in_place=1
tidb >begin optimistic;
tidb >insert into t values (1);
ERROR 1062 : Duplicate entry '1' for key 'PRIMARY'
```

Constraint checking is always performed in place for pessimistic transactions (default).

tidb_check_mb4_value_in_utf8

- Scope: SERVER
- Default value: 1, indicating check the validity of UTF-8 data. This default behavior is compatible with MySQL.
- This variable is used to set whether to check the validity of UTF-8 data.
- To upgrade an earlier version (TiDB v2.1.1 or earlier), you may need to disable this option. Otherwise, you can successfully write invalid strings in an earlier version but fail to do this in a later version, because there is no data validity check in the earlier version. For details, see FAQs After Upgrade.

tidb_opt_insubq_to_join_and_agg

- Scope: SESSION | GLOBAL
- Default value: 1
- This variable is used to set whether to enable the optimization rule that converts a subquery to join and aggregation.
- For example, after you enable this optimization rule, the subquery is converted as follows:

```
select * from t where t.a in (select aa from t1)
```

  The subquery is converted to join as follows:

```
select * from t, (select aa from t1 group by aa) tmp_t where t.a =
    ↪ tmp_t.aa
```

  If `t1` is limited to be `unique` and `not null` in the `aa` column. You can use the following statement, without aggregation.

```
select * from t, t1 where t.a=t1.a
```

tidb_opt_correlation_threshold

- Scope: SESSION | GLOBAL
- Default value: 0.9
- This variable is used to set the threshold value that determines whether to enable estimating the row count by using column order correlation. If the order correlation between the current column and the `handle` column exceeds the threshold value, this method is enabled.

tidb_opt_correlation_exp_factor

- Scope: SESSION | GLOBAL
- Default value: 1
- When the method that estimates the number of rows based on column order correlation is not available, the heuristic estimation method is used. This variable is used to control the behavior of the heuristic method.
  - When the value is 0, the heuristic method is not used.
  - When the value is greater than 0:
    * A larger value indicates that an index scan will probably be used in the heuristic method.
    * A smaller value indicates that a table scan will probably be used in the heuristic method.

tidb_enable_window_function

- Scope: SESSION | GLOBAL
- Default value: 1, indicating enabling the window function feature.
- This variable is used to control whether to enable the support for window functions. Note that window functions may use reserved keywords. This might cause SQL statements that could be executed normally cannot be parsed after upgrading TiDB. In this case, you can set `tidb_enable_window_function` to `0`.

tidb_slow_query_file

- Scope: SESSION
- Default value: “”
- When `INFORMATION_SCHEMA.SLOW_QUERY` is queried, only the slow query log name set by `slow-query-file` in the configuration file is parsed. The default slow query log name is “tidb-slow.log”. To parse other logs, set the `tidb_slow_query_file` session variable to a specific file path, and then query `INFORMATION_SCHEMA.SLOW_QUERY` to parse the slow query log based on the set file path. For details, see Identify Slow Queries.

tidb_enable_fast_analyze

- Scope: SESSION | GLOBAL
- Default value: 0, indicating not enabling the statistics fast `Analyze` feature.
- This variable is used to set whether to enable the statistics `Fast Analyze` feature.
- If the statistics `Fast Analyze` feature is enabled, TiDB randomly samples about 10,000 rows of data as statistics. When the data is distributed unevenly or the data size is small, the statistics accuracy is low. This might lead to a non-optimal execution plan, for example, selecting a wrong index. If the execution time of the regular `Analyze` statement is acceptable, it is recommended to disable the `Fast Analyze` feature.

tidb_expensive_query_time_threshold

- Scope: SERVER
- Default value: 60
- This variable is used to set the threshold value that determines whether to print expensive query logs. The unit is second. The difference between expensive query logs and slow query logs is:
  - Slow logs are printed after the statement is executed.
  - Expensive query logs print the statements that are being executed, with execution time exceeding the threshold value, and their related information.

tidb_wait_split_region_finish

- Scope: SESSION
- Default value: 1, indicating returning the result after all Regions are scattered.
- It usually takes a long time to scatter Regions, which is determined by PD scheduling and TiKV loads. This variable is used to set whether to return the result to the client after all Regions are scattered completely when the `SPLIT REGION` statement is being executed. Value `0` indicates returning the value before finishing scattering all Regions.
- Note that when scattering Regions, the write and read performances for the Region that is being scattered might be affected. In batch-write or data importing scenarios, it is recommended to import data after Regions scattering is finished.

tidb_wait_split_region_timeout

- Scope: SESSION
- Default value: 300
- This variable is used to set the timeout for executing the `SPLIT REGION` statement. The unit is second. If a statement is not executed completely within the specified time value, a timeout error is returned.

tidb_scatter_region

- Scope: GLOBAL

- Default value: 0
- By default, Regions are split for a new table when it is being created in TiDB. After this variable is enabled, the newly split Regions are scattered immediately during the execution of the `CREATE TABLE` statement. This applies to the scenario where data need to be written in batches right after the tables are created in batches, because the newly split Regions can be scattered in TiKV beforehand and do not have to wait to be scheduled by PD. To ensure the continuous stability of writing data in batches, the `CREATE TABLE` statement returns success only after the Regions are successfully scattered. This makes the statement's execution time multiple times longer than that when you disable this variable.

tidb_allow_remove_auto_inc New in v3.0.4

- Scope: SESSION
- Default value: 0
- This variable is used to set whether the `AUTO_INCREMENT` property of a column is allowed to be removed by executing `ALTER TABLE MODIFY` or `ALTER TABLE CHANGE` statements. It is not allowed by default.

tidb_enable_stmt_summary New in v3.0.4

- Scope: SESSION | GLOBAL
- Default value: 0
- This variable is used to enable or disable the statement summary feature. If enabled, SQL execution information like time consumption is recorded to the `performance_schema.events_statements_summary_by_digest` table to identify and troubleshoot SQL performance issues.

4.2.1.3 Configuration Options

TiDB is configurable using command-line options and environment variables. The default TiDB ports are 4000 for client requests and 10080 for status report.

4.2.1.3.1 `--advertise-address`

- The IP address through which to log into the TiDB server
- Default: “”
- This address must be accessible by the rest of the TiDB cluster and the user.

4.2.1.3.2 `--binlog-socket`

- The TiDB services use the unix socket file for internal connections, such as the Pump service
- Default: “”
- You can use “/tmp/pump.sock” to accept the communication of Pump unix socket file.

4.2.1.3.3 `--config`

- The configuration file
- Default: ""
- If you have specified the configuration file, TiDB reads the configuration file. If the corresponding configuration also exists in the command line options, TiDB uses the configuration in the command line options to overwrite that in the configuration file. For detailed configuration information, see TiDB Configuration File Description.

4.2.1.3.4 `--cors`

- Specifies the `Access-Control-Allow-Origin` value for Cross-Origin Request Sharing (CORS) request of the TiDB HTTP status service
- Default: ""

4.2.1.3.5 `--host`

- The host address that the TiDB server monitors
- Default: "0.0.0.0"
- The TiDB server monitors this address.
- The "0.0.0.0" address monitors all network cards by default. If you have multiple network cards, specify the network card that provides service, such as `192.168.100.113`.

4.2.1.3.6 `-L`

- The log level
- Default: "info"
- You can choose from "debug", "info", "warn", "error", or "fatal".

4.2.1.3.7 `--log-file`

- The log file
- Default: ""
- If this option is not set, logs are output to "stderr". If this option is set, logs are output to the corresponding file, which is automatically rotated in the early morning every day, and the previous file is renamed as a backup.

4.2.1.3.8 `--log-slow-query`

- The directory for the slow query log
- Default: ""
- If this option is not set, logs are output to the file specified by `--log-file` by default.

4.2.1.3.9 `--metrics-addr`

- The Prometheus Pushgateway address
- Default: “”
- Leaving it empty stops the Prometheus client from pushing.
- The format is `--metrics-addr=192.168.100.115:9091`.

4.2.1.3.10 `--metrics-interval`

- The Prometheus client push interval in seconds
- Default: 15s
- Setting the value to 0 stops the Prometheus client from pushing.

4.2.1.3.11 `-P`

- The monitoring port of TiDB services
- Default: “4000”
- The TiDB server accepts MySQL client requests from this port.

4.2.1.3.12 `--path`

- The path to the data directory for local storage engine like “mocktikv”
- For `--store = tikv`, you must specify the path; for `--store = mocktikv`, the default value is used if you do not specify the path.
- For the distributed storage engine like TiKV, `--path` specifies the actual PD address. Assuming that you deploy the PD server on 192.168.100.113:2379, 192.168.100.114:2379 and 192.168.100.115:2379, the value of `--path` is “192.168.100.113:2379, 192.168.100.114:2379, 192.168.100.115:2379”.
- Default: “/tmp/tidb”
- You can use `tidb-server --store=mocktikv --path=""` to enable a pure in-memory TiDB.

4.2.1.3.13 `--proxy-protocol-networks`

- The list of proxy server’s IP addresses allowed by PROXY Protocol; if you need to configure multiple addresses, separate them using “,”.
- Default: “”
- Leaving it empty disables PROXY Protocol. The value can be the IP address (192.168.1.50) or CIDR (192.168.1.0/24). “*” means any IP addresses.

4.2.1.3.14 `--proxy-protocol-header-timeout`

- Timeout for the PROXY protocol header read
- Default: 5 (seconds)

> **Note:**
> Do not set the value to `0`. Use the default value except for special situations.

4.2.1.3.15 `--report-status`

- Enables (`true`) or disables (`false`) the status report and pprof tool
- Default: `true`
- When set to `true`, this parameter enables metrics and pprof. When set to `false`, this parameter disables metrics and pprof.

4.2.1.3.16 `--run-ddl`

- To see whether the `tidb-server` runs DDL statements, and set when the number of `tidb-server` is over two in the cluster
- Default: `true`
- The value can be (true) or (false). (true) indicates the `tidb-server` runs DDL itself. (false) indicates the `tidb-server` does not run DDL itself.

4.2.1.3.17 `--socket string`

- The TiDB services use the unix socket file for external connections.
- Default: “”
- Use `/tmp/tidb.sock` to open the unix socket file.

4.2.1.3.18 `--status`

- The status report port for TiDB server
- Default: “10080”
- This port is used to get server internal data. The data includes Prometheus metrics and pprof.
- Prometheus metrics can be accessed by `"http://host:status_port/metrics"`.
- pprof data can be accessed by `"http://host:status_port/debug/pprof"`.

4.2.1.3.19 `--status-host`

- The `HOST` used to monitor the status of TiDB service
- Default: `0.0.0.0`

4.2.1.3.20 `--store`

- Specifies the storage engine used by TiDB in the bottom layer
- Default: "mocktikv"
- You can choose "mocktikv" or "tikv". ("mocktikv" is the local storage engine; "tikv" is a distributed storage engine)

4.2.1.3.21 `--token-limit`

- The number of sessions allowed to run concurrently in TiDB. It is used for traffic control.
- Default: 1000
- If the number of the concurrent sessions is larger than `token-limit`, the request is blocked and waiting for the operations which have been finished to release tokens.

4.2.1.3.22 `-V`

- Outputs the version of TiDB
- Default: ""

4.2.1.4 TiDB Configuration File

The TiDB configuration file supports more options than command-line parameters. You can download the default configuration file `config.toml.example` and rename it to `config.toml`. This document describes only the options that are not involved in command line options.

`split-table`

- Determines whether to create a separate Region for each table
- Default value: `true`
- It is recommended to set it to `false` if you need to create a large number of tables.

`oom-action`

- Specifies the operation when out-of-memory occurs in TiDB
- Default value: `log`

- The valid options are `log` and `cancel`. `log` only prints the log without actual processing. `cancel"` cancels the operation and outputs the log.

`mem-quota-query`

- The maximum memory available for a single SQL statement
- Default value: `34359738368`
- Requests that require more memory than this value are handled based on the behavior defined by `oom-action`.

`lower-case-table-names`

- Configures the value of the `lower-case-table-names` system variable.
- Default value: `2`
- For details, see the MySQL description of this variable.

> **Note:**
> Currently, TiDB only supports setting the value of this option to `2`. This means it is case-sensitive when you save a table name, but case-insensitive when you compare table names. The comparison is based on the lower case.

`lease`

- The timeout of the DDL lease
- Default value: `45s`
- Unit: second

`compatible-kill-query`

- Determines whether to set the `KILL` statement to be MySQL compatible
- Default value: `false`
- The behavior of `KILL xxx` in TiDB differs from the behavior in MySQL. TiDB requires the `TIDB` keyword, namely, `KILL TIDB xxx`. If `compatible-kill-query` is set to `true`, the `TIDB` keyword is not needed.
- This distinction is important because the default behavior of the MySQL command-line client, when the user hits Ctrl+C, is to create a new connection to the backend and execute the `KILL` statement in that new connection. If a load balancer or proxy has sent the new connection to a different TiDB server instance than the original session, the wrong session could be terminated, which could cause interruption to applications using the cluster. Enable `compatible-kill-query` only if you are certain that the connection you refer to in your `KILL` statement is on the same server to which you send the `KILL` statement.

`check-mb4-value-in-utf8`

- Determines whether to enable the `utf8mb4` character check. When this feature is enabled, if the character set is `utf8` and the `mb4` characters are inserted in `utf8`, an error is returned.
- Default value: `false`

`treat-old-version-utf8-as-utf8mb4`

- Determines whether to treat the `utf8` character set in old tables as `utf8mb4`
- Default value: `true`

`alter-primary-key`

- Determines whether to add or remove the primary key constraint to or from a column.
- Default value: `false`
- With this default setting, adding or removing the primary key constraint is not supported. You can enable this feature by setting `alter-primary-key` to `true`. However, if a table already exists before the switch is on, and the data type of its primary key column is an integer, dropping the primary key from the column is not possible even if you set this configuration item to `true`.

`server-version`

- Modifies the version string returned by TiDB in the following situations:
  - When the built-in `VERSION()` function is used.
  - When TiDB establishes the initial connection to the client and returns the initial handshake packet with version string of the server. For details, see MySQL Initial Handshake Packet.
- Default value: ""
- By default, the format of the TiDB version string is `5.7.${mysql_latest_minor_version}-TiDB-${tidb_version}`.

`max-index-length`

- Sets the maximum allowable length of the newly created index.
- Default value: `3072`
- Unit: byte
- Currently, the valid value range is `[3072, 3072*4]`. MySQL and TiDB (version < v3.0.11) do not have this configuration item, but both limit the length of the newly created index. This limit in MySQL is `3072`. In TiDB (version =< 3.0.7), this limit is `3072*4`. In TiDB (3.0.7 < version < 3.0.11), this limit is `3072`. This configuration is added to be compatible with MySQL and earlier versions of TiDB.

4.2.1.4.1 Log

Configuration items related to log

`format`

- Specifies the log output format
- Available values: `json`, `text`, `console`
- Default value: `text`

`disable-timestamp`

- Determines whether to disable timestamp output in the log
- Default value: `false`
- If you set the value to `true`, the log does not output timestamp

`slow-query-file`

- The file name of the slow query log
- Default value: `tidb-slow.log`
- The format of the slow log is updated in TiDB v2.1.8, so the slow log is output to the slow log file separately. In versions before v2.1.8, this variable is set to "" by default.
- After you set it, the slow query log is output to this file separately

`slow-threshold`

- Outputs the threshold value of consumed time in the slow log
- Default value: `300ms`
- If the value in a query is larger than the default value, it is a slow query and is output to the slow log

`expensive-threshold`

- Outputs the threshold value of the number of rows for the `expensive` operation
- Default value: `10000`
- When the number of query rows (including the intermediate results based on statistics) is larger than this value, it is an `expensive` operation and outputs log with the `[ ↪ EXPENSIVE_QUERY]` prefix.

`query-log-max-len`

- The maximum length of SQL output
- Default value: `2048`
- When the length of the statement is longer than `query-log-max-len`, the statement is truncated to output.

4.2.1.4.2 log.file

Configuration items related to log files

`filename`

- The file name of the general log file
- Default value: ""
- If you set it, the log is output to this file

`max-size`

- The size limit of the log file
- Default value: 300MB
- The maximum size is 4GB.

`max-days`

- The maximum number of days that the log is retained
- Default value: `0`
- The log is retained by default. If you set the value, the expired log is cleaned up after `max-days`.

`max-backups`

- The maximum number of retained logs
- Default value: `0`
- All the log files are retained by default. If you set it to `7`, seven log files are retained at maximum.

`log-rotate`

- Determines whether to create a new log file every day
- Default value: `true`
- If you set the parameter to `true`, a new log file is created every day. If you set it to `false`, the log is output to a single log file.

4.2.1.4.3 Security

Configuration items related to security

`ssl-ca`

- The file path of the trusted CA certificate in the PEM format
- Default value: ""

- If you set this option and `--ssl-cert`, `--ssl-key` at the same time, TiDB authenticates the client certificate based on the list of trusted CAs specified by this option when the client presents the certificate. If the authentication fails, the connection is terminated.
- If you set this option but the client does not present the certificate, the secure connection continues without client certificate authentication.

`ssl-cert`

- The file path of the SSL certificate in the PEM format
- Default value: “”
- If you set this option and `--ssl-key` at the same time, TiDB allows (but not forces) the client to securely connect to TiDB using TLS
- If the specified certificate or private key is invalid, TiDB starts as usual but cannot receive secure connection

`ssl-key`

- The file path of the SSL certificate key in the PEM format, that is the private key of the certificate specified by `--ssl-cert`
- Default value: “”
- Currently, TiDB does not support loading the private keys protected by passwords

`cluster-ssl-ca`

- The CA root certificate used to connect TiKV or PD with TLS
- Default value: “”

`cluster-ssl-cert`

- The path of the SSL certificate file used to connect TiKV or PD with TLS
- Default value: “”

`cluster-ssl-key`

- The path of the SSL private key file used to connect TiKV or PD with TLS
- Default value: “”

`skip-grant-table`

- Determines whether to skip permission check
- Default value: `false`

4.2.1.4.4 Performance

Configuration items related to performance

`max-procs`

- The number of CPUs used by TiDB
- Default value: `0`
- The default `0` indicates using all the CPUs on the machine. You can also set it to n, and then TiDB uses n CPUs.

`max-memory`

- The maximum memory limit for the Prepared Least Recently Used (LRU) caching. If this value exceeds `performance.max-memory * (1 - prepared-plan-cache.memory -guard-ratio)`, the elements in the LRU are removed.
- Default value: `0`
- This configuration only takes effect when `prepared-plan-cache.enabled` is `true`. When the size of the LRU is greater than `prepared-plan-cache.capacity`, the elements in the LRU are also removed.

`stmt-count-limit`

- The maximum number of statements allowed in a single TiDB transaction
- Default value: `5000`
- If a transaction does not roll back or commit after the number of statements exceeds `stmt-count-limit`, TiDB returns the `statement count 5001 exceeds the transaction limitation, autocommit = false` error. This configuration takes effect **only** in the optimistic transaction. If you use the pessimistic transaction, the number of statements in a transaction is not limited by this configuration.

`tcp-keep-alive`

- Determines whether to enable `keepalive` in the TCP layer
- Default value: `true`

`cross-join`

- Default value: `true`
- TiDB supports executing the `JOIN` statement without any condition (the `WHERE` field) of both sides tables by default; if you set the value to `false`, the server refuses to execute when such a `JOIN` statement appears.

`stats-lease`

- The time interval of reloading statistics, updating the number of table rows, checking whether it is needed to perform the automatic analysis, using feedback to update statistics and loading statistics of columns
- Default value: `3s`
  - At intervals of `stats-lease` time, TiDB checks the statistics for updates and updates them to the memory if updates exist
  - At intervals of `20 * stats-lease` time, TiDB updates the total number of rows generated by DML and the number of modified rows to the system table
  - At intervals of `stats-lease`, TiDB checks for tables and indexes that need to be automatically analyzed
  - At intervals of `stats-lease`, TiDB checks for column statistics that need to be loaded to the memory
  - At intervals of `200 * stats-lease`, TiDB writes the feedback cached in the memory to the system table
  - At intervals of `5 * stats-lease`, TiDB reads the feedback in the system table, and updates the statistics cached in the memory
- When `stats-lease` is set to 0s, TiDB periodically reads the feedback in the system table, and updates the statistics cached in the memory every three seconds. But TiDB no longer automatically modifies the following statistics-related system tables:
  - `mysql.stats_meta`: TiDB no longer automatically records the number of table rows that are modified by the transaction and updates it to this system table
  - `mysql.stats_histograms`/`mysql.stats_buckets` and `mysql.stats_top_n`: TiDB no longer automatically analyzes and proactively updates statistics
  - `mysql.stats_feedback`: TiDB no longer updates the statistics of the tables and indexes according to a part of statistics returned by the queried data

`run-auto-analyze`

- Determines whether TiDB executes automatic analysis
- Default value: `true`

`feedback-probability`

- The probability that TiDB collects the feedback statistics of each query
- Default value: `0.05`
- TiDB collects the feedback of each query at the probability of `feedback-probability`, to update statistics

`query-feedback-limit`

- The maximum pieces of query feedback that can be cached in memory. Extra pieces of feedback that exceed this limit are discarded.

- Default value: `1024`

`pseudo-estimate-ratio`

- The ratio of (number of modified rows)/(total number of rows) in a table. If the value is exceeded, the system assumes that the statistics have expired and the pseudo statistics will be used.
- Default value: `0.8`
- The minimum value is `0` and the maximum value is `1`.

`force-priority`

- Sets the priority for all statements
- Default: `NO_PRIORITY`
- Optional values: `NO_PRIORITY`, `LOW_PRIORITY`, `HIGH_PRIORITY`, `DELAYED`

4.2.1.4.5 prepared-plan-cache

The Plan Cache configuration of the `PREPARE` statement

> **Warning:**
>
> This is still an experimental feature. It is recommended **not** to use this feature in the production environment.

`enabled`

- Determines whether to enable Plan Cache of the `PREPARE` statement
- Default value: `false`

`capacity`

- The number of cached statements
- Default value: `100`

`memory-guard-ratio`

- It is used to prevent `performance.max-memory` from being exceeded. When `max-proc` ↪ `* (1 - prepared-plan-cache.memory-guard-ratio)` is exceeded, the elements in the LRU are removed.
- Default value: `0.1`
- The minimum value is `0`; the maximum value is `1`

4.2.1.4.6 tikv-client

`grpc-connection-count`

- The maximum number of connections established with each TiKV
- Default value: `16`

`grpc-keepalive-time`

- The `keepalive` time interval of the RPC connection between TiDB and TiKV nodes. If there is no network packet within the specified time interval, the gRPC client executes `ping` command to TiKV to see if it is alive.
- Default: `10`
- unit: second

`grpc-keepalive-timeout`

- The timeout of the RPC `keepalive` check between TiDB and TiKV nodes
- Default value: `3`
- unit: second

`commit-timeout`

- The maximum timeout when executing a transaction commit
- Default value: `41s`
- It is required to set this value larger than twice of the Raft election timeout.

`max-txn-time-use`

- The maximum execution time allowed for a single transaction
- Default value: `590`
- unit: second

`max-batch-size`

- The maximum number of RPC packets sent in batch. If the value is not `0`, the `BatchCommands` API is used to send requests to TiKV, and the RPC latency can be reduced in the case of high concurrency. It is recommended that you do not modify this value.
- Default value: `128`

`max-batch-wait-time`

- Waits for `max-batch-wait-time` to encapsulate the data packets into a large packet in batch and send it to the TiKV node. It is valid only when the value of `tikv-client.max-batch-size` is greater than `0`. It is recommended not to modify this value.
- Default value: `0`
- unit: nanoseconds

`batch-wait-size`

- The maximum number of packets sent to TiKV in batch. It is recommended not to modify this value.
- Default value: `8`
- If the value is `0`, this feature is disabled.

`overload-threshold`

- The threshold of the TiKV load. If the TiKV load exceeds this threshold, more `batch` packets are collected to relieve the pressure of TiKV. It is valid only when the value of `tikv-client.max-batch-size` is greater than `0`. It is recommended not to modify this value.
- Default value: `200`

txn-local-latches

Configuration related to the transaction latch. It is recommended to enable it when many local transaction conflicts occur.

`enable`

- Determines whether to enable the memory lock of transactions
- Default value: `false`

`capacity`

- The number of slots corresponding to Hash, which automatically adjusts upward to an exponential multiple of 2. Each slot occupies 32 Bytes of memory. If set too small, it might result in slower running speed and poor performance in the scenario where data writing covers a relatively large range (such as importing data).
- Default value: `2048000`

4.2.1.4.7 binlog

Configurations related to TiDB Binlog

`enable`

- Enables or disables binlog
- Default value: `false`

`write-timeout`

- The timeout of writing binlog into Pump. It is not recommended to modify this value.
- Default: `15s`
- unit: second

`ignore-error`

- Determines whether to ignore errors occurred in the process of writing binlog into Pump. It is not recommended to modify this value.
- Default value: `false`
- When the value is set to `true` and an error occurs, TiDB stops writing binlog and add `1` to the count of the `tidb_server_critical_error_total` monitoring item. When the value is set to `false`, the binlog writing fails and the entire TiDB service is stopped.

`binlog-socket`

- The network address to which binlog is exported
- Default value: ""

`strategy`

- The strategy of Pump selection when binlog is exported. Currently, only the `hash` and `range` methods are supported.
- Default value: `range`

4.2.1.4.8 status

Configuration related to the status of TiDB service

`report-status`

- Enables or disables the HTTP API service
- Default value: true

`record-db-qps`

- Determines whether to transmit the database-related QPS metrics to Prometheus
- Default value: `false`

4.2.1.4.9 stmt-summary New in v3.0.4

Configurations related to the `events_statement_summary_by_digest` table

max-stmt-count

- The maximum number of SQL categories allowed to be saved in the `events_statement_summary_by_` table
- Default value: 100

max-sql-length

- The longest display length for the `DIGEST_TEXT` and `QUERY_SAMPLE_TEXT` columns in the `events_statement_summary_by_digest` table.
- Default value: `4096`

4.2.1.4.10 pessimistic-txn

enable

- Enables the pessimistic transaction mode. For pessimistic transaction usage, refer to TiDB Pessimistic Transaction Mode.
- Default value: `true`

max-retry-count

- The max number of retries of each statement in pessimistic transactions. Exceeding this limit results in error.
- Default value: `256`

4.2.2 pd-server

4.2.2.1 PD Configuration Flags

PD is configurable using command-line flags and environment variables.

4.2.2.1.1 `--advertise-client-urls`

- The list of advertise URLs for the client to access PD
- Default: `"${client-urls}"`
- In some situations such as in the Docker or NAT network environment, if a client cannot access PD through the default client URLs listened to by PD, you must manually set the advertise client URLs.
- For example, the internal IP address of Docker is `172.17.0.1`, while the IP address of the host is `192.168.100.113` and the port mapping is set to `-p 2379:2379`. In this case, you can set `--advertise-client-urls` to `"http://192.168.100.113:2379"`. The client can find this service through `"http://192.168.100.113:2379"`.

4.2.2.1.2 `--advertise-peer-urls`

- The list of advertise URLs for other PD nodes (peers) to access a PD node
- Default: `"${peer-urls}"`
- In some situations such as in the Docker or NAT network environment, if the other nodes (peers) cannot access the PD node through the default peer URLs listened to by this PD node, you must manually set the advertise peer URLs.
- For example, the internal IP address of Docker is `172.17.0.1`, while the IP address of the host is `192.168.100.113` and the port mapping is set to `-p 2380:2380`. In this case, you can set `--advertise-peer-urls` to `"http://192.168.100.113:2380"`. The other PD nodes can find this service through `"http://192.168.100.113:2380"`.

4.2.2.1.3 `--client-urls`

- The list of client URLs to be listened to by PD
- Default: `"http://127.0.0.1:2379"`
- When you deploy a cluster, you must specify the IP address of the current host as `--client-urls` (for example, `"http://192.168.100.113:2379"`). If the cluster runs on Docker, specify the IP address of Docker as `"http://0.0.0.0:2379"`.

4.2.2.1.4 `--peer-urls`

- The list of peer URLs to be listened to by a PD node
- Default: `"http://127.0.0.1:2380"`
- When you deploy a cluster, you must specify `--peer-urls` as the IP address of the current host, such as `"http://192.168.100.113:2380"`. If the cluster runs on Docker, specify the IP address of Docker as `"http://0.0.0.0:2380"`.

4.2.2.1.5 `--config`

- The configuration file
- Default: “”
- If you set the configuration using the command line, the same setting in the configuration file will be overwritten.

4.2.2.1.6 `--data-dir`

- The path to the data directory
- Default: “default.${name}”

4.2.2.1.7 `--initial-cluster`

- The initial cluster configuration for bootstrapping
- Default: `"{name}=http://{advertise-peer-url}"`
- For example, if `name` is "pd", and `advertise-peer-urls` is `"http://192.168.100.113:2380"` ↪ , the `initial-cluster` is `"pd=http://192.168.100.113:2380"`.
- If you need to start three PD servers, the `initial-cluster` might be:

```
pd1=http://192.168.100.113:2380, pd2=http://192.168.100.114:2380, pd3
    ↪ =192.168.100.115:2380
```

4.2.2.1.8 `--join`

- Join the cluster dynamically
- Default: ""
- If you want to join an existing cluster, you can use `--join="${advertise-client-` ↪ `urls}"`, the `advertise-client-url` is any existing PD's, multiply advertise client urls are separated by comma.

4.2.2.1.9 `-L`

- The log level
- Default: "info"
- You can choose from debug, info, warn, error, or fatal.

4.2.2.1.10 `--log-file`

- The log file
- Default: ""
- If this flag is not set, logs will be written to stderr. Otherwise, logs will be stored in the log file which will be automatically rotated every day.

4.2.2.1.11 `--log-rotate`

- To enable or disable log rotation
- Default: true
- When the value is true, follow the `[log.file]` in PD configuration files.

4.2.2.1.12 `--name`

- The human-readable unique name for this PD member
- Default: "pd"
- If you want to start multiply PDs, you must use different name for each one.

4.2.2.1.13 `--cacert`

- The file path of CA, used to enable TLS
- Default: ""

4.2.2.1.14 `--cert`

- The path of the PEM file including the X509 certificate, used to enable TLS
- Default: ""

4.2.2.1.15 `--key`

- The path of the PEM file including the X509 key, used to enable TLS
- Default: ""

4.2.2.1.16 `--namespace-classifier`

- To specify the namespace classifier used by PD
- Default: "table"
- If you use TiKV separately, not in the entire TiDB cluster, it is recommended to configure the value to 'default'.

4.2.2.1.17 `--metrics-addr`

- The address of Prometheus Pushgateway, which does not push data to Prometheus by default.
- Default: ""

4.2.2.2 PD Configuration File

The PD configuration file supports more options than command-line parameters. You can find the default configuration file here.

This document only describes parameters that are not included in command-line parameters. Check here for the command line parameters.

`lease`

- The timeout of the PD Leader Key lease. After the timeout, the system re-elects a Leader.
- Default value: `3`
- unit: second

`tso-save-interval`

- The interval for PD to allocate TSOs for persistent storage in etcd
- Default value: `3` seconds

`initial-cluster-state`

- The initial state of the cluster
- Default value: `new`

`enable-prevote`

- Enables or disables `raft prevote`
- Default value: `true`

`quota-backend-bytes`

- The storage size of the meta-information database, which is 2GB by default
- Default value: `2147483648`

`auto-compaction-mod`

- The automatic compaction modes of the meta-information database
- Available options: `periodic` (by cycle) and `revision` (by version number).
- Default value: `periodic`

`auto-compaction-retention`

- The time interval for automatic compaction of the meta-information database when `auto-compaction-retention` is `periodic`. When the compaction mode is set to `revision`, this parameter indicates the version number for the automatic compaction.
- Default value: 1h

`force-new-cluster`

- Determines whether to force PD to start as a new cluster and modify the number of Raft members to `1`

- Default value: `false`

`tick-interval`

- The tick period of etcd Raft
- Default value: `100ms`

`election-interval`

- The timeout for the etcd leader election
- Default value: `3s`

`use-region-storage`

- Enables or disables independent Region storage
- Default value: `false`

4.2.2.2.1 security

Configuration items related to security

`cacert-path`

- The path of the CA file
- Default value: ""

`cert-path`

- The path of the Privacy Enhanced Mail (PEM) file that contains the X509 certificate
- Default value: ""

`key-path`

- The path of the PEM file that contains the X509 key
- Default value: ""

4.2.2.2.2 `log`

Configuration items related to log

`format`

- The log format, which can be specified as "text", "json", or "console"
- Default value: `text`

`disable-timestamp`

- Whether to disable the automatically generated timestamp in the log
- Default value: `false`

4.2.2.2.3 `log.file`

Configuration items related to the log file

`max-size`

- The maximum size of a single log file. When this value is exceeded, the system automatically splits the log into several files.
- Default value: `300`
- Unit: MiB
- Minimum value: `1`

`max-days`

- The maximum number of days in which a log is kept
- Default value: `28`
- Minimum value: `1`

`max-backups`

- The maximum number of log files to keep
- Default value: `7`
- Minimum value: `1`

4.2.2.2.4 `metric`

Configuration items related to monitoring

`interval`

- The interval at which monitoring metric data is pushed to Prometheus
- Default value: `15s`

4.2.2.2.5 `schedule`

Configuration items related to scheduling

`max-merge-region-size`

- Controls the size limit of `Region Merge`. When the Region size is greater than the specified value, PD does not merge the Region with the adjacent Regions.
- Default value: `20`

`max-merge-region-keys`

- Specifies the upper limit of the `Region Merge` key. When the Region key is greater than the specified value, the PD does not merge the Region with its adjacent Regions.
- Default value: `200000`

`patrol-region-interval`

- Controls the running frequency at which `replicaChecker` checks the health state of a Region. The smaller this value is, the faster `replicaChecker` runs. Normally, you do not need to adjust this parameter.
- Default value: `100ms`

`split-merge-interval`

- Controls the time interval between the `split` and `merge` operations on the same Region. That means a newly split Region will not be merged for a while.
- Default value: `1h`

`max-snapshot-count`

- Control the maximum number of snapshots that a single store receives or sends at the same time. PD schedulers depend on this configuration to prevent the resources used for normal traffic from being preempted.
- Default value value: `3`

`max-pending-peer-count`

- Controls the maximum number of pending peers in a single store. PD schedulers depend on this configuration to prevent too many Regions with outdated logs from being generated on some nodes.
- Default value: `16`

`max-store-down-time`

- The downtime after which PD judges that the disconnected store can not be recovered. When PD fails to receive the heartbeat from a store after the specified period of time, it adds replicas at other nodes.
- Default value: `30m`

`leader-schedule-limit`

- The number of Leader scheduling tasks performed at the same time
- Default value: `4`

`region-schedule-limit`

- The number of Region scheduling tasks performed at the same time
- Default value: `4`

`replica-schedule-limit`

- The number of Replica scheduling tasks performed at the same time
- Default value: `8`

`merge-schedule-limit`

- The number of the `Region Merge` scheduling tasks performed at the same time. Set this parameter to `0` to disable `Region Merge`.
- Default value: `8`

`high-space-ratio`

- The threshold ratio below which the capacity of the store is sufficient
- Default value: `0.6`
- Minimum value: greater than `0`
- Maximum value: less than `1`

`low-space-ratio`

- The threshold ratio above which the capacity of the store is insufficient
- Default value: `0.8`
- Minimum value: greater than `0`
- Maximum value: less than `1`

`tolerant-size-ratio`

- Controls the `balance` buffer size
- Default value: `5`
- Minimum value: `0`

`disable-remove-down-replica`

- Determines whether to disable the feature that automatically removes `DownReplica`. When this parameter is set to `true`, PD does not automatically clean up the copy in the down state.
- Default value: `false`

`disable-replace-offline-replica`

- Determines whether to disable the feature that migrates `OfflineReplica`. When this parameter is set to `true`, PD does not migrate the replicas in the offline state.
- Default value: `false`

`disable-make-up-replica`

- Determines whether to disable the feature that automatically supplements replicas. When this parameter is set to `true`, PD does not supplement replicas for the Region with insufficient replicas.
- Default value: `false`

`disable-remove-extra-replica`

- Determines whether to disable the feature that removes extra replicas. When this parameter is set to `true`, PD does not remove the extra replicas from the Region with excessive replicas.
- Default value: `false`

`disable-location-replacement`

- Determines whether to disable isolation level check. When this parameter is set to `true` ↪ , PD does not increase the isolation level of the Region replicas through scheduling.
- Default value: `false`

4.2.2.2.6 `replication`

Configuration items related to replicas

`max-replicas`

- The number of replicas
- Default value: `3`

`location-labels`

- The topology information of a TiKV cluster
- Default value: `[]`

4.2.2.2.7 `label-property`

Configuration items related to labels

`key`

- The label key for the store that rejected the Leader
- Default value: `""`

`value`

- The label value for the store that rejected the Leader
- Default value: `""`

4.2.3 tikv-server

4.2.3.1 TiKV Configuration Flags

TiKV supports some readable unit conversions for command line parameters.

- File size (based on byte): KB, MB, GB, TB, PB (or lowercase)
- Time (based on ms): ms, s, m, h

4.2.3.1.1 `-A, --addr`

- The address that the TiKV server monitors
- Default: “127.0.0.1:20160”
- To deploy a cluster, you must use `--addr` to specify the IP address of the current host, such as “192.168.100.113:20160”. If the cluster is run on Docker, specify the IP address of Docker as “0.0.0.0:20160”.

4.2.3.1.2 `--advertise-addr`

- The server advertise address for client traffic from outside
- Default: ${addr}
- If the client cannot connect to TiKV through the default monitoring address because of Docker or NAT network, you must manually set the advertise address explicitly.
- For example, the internal IP address of Docker is 172.17.0.1, while the IP address of the host is 192.168.100.113 and the port mapping is set to `-p 20160:20160`. In this case, you can set `--advertise-addr` to “192.168.100.113:20160”. The client can find this service through 192.168.100.113:20160.

4.2.3.1.3 `-C, --config`

- The config file
- Default: “”
- If you set the configuration using the command line, the same setting in the config file will be overwritten.

4.2.3.1.4 `--capacity`

- The store capacity
- Default: 0 (unlimited)
- PD uses this flag to determine how to balance the TiKV servers. (Tip: you can use 10GB instead of 1073741824)

4.2.3.1.5 `--data-dir`

- The path to the data directory
- Default: “/tmp/tikv/store”

4.2.3.1.6 `-L`

- The log level
- Default: “info”
- You can choose from trace, debug, info, warn, error, or off.

4.2.3.1.7 `--log-file`

- The log file
- Default: “”
- If this flag is not set, logs will be written to stderr. Otherwise, logs will be stored in the log file which will be automatically rotated every day.

4.2.3.1.8 `--pd`

- The address list of PD servers
- Default: “”
- To make TiKV work, you must use the value of `--pd` to connect the TiKV server to the PD server. Separate multiple PD addresses using comma, for example “192.168.100.113:2379, 192.168.100.114:2379, 192.168.100.115:2379”.

— title: TiKV Configuration File summary: Learn the TiKV configuration file. aliases: [‘/docs/v3.0/tikv-configuration-file/’,‘/docs/v3.0/reference/configuration/tikv-server/configuration-file/’] —

4.2.3.2 TiKV Configuration File

The TiKV configuration file supports more options than command-line parameters. You can find the default configuration file in etc/config-template.toml and rename it to `config.toml`.

This document only describes the parameters that are not included in command-line parameters. For more details, see command-line parameter.

server

- Configuration items related to the server

4.2.3.2.1 `status-thread-pool-size`

- The number of worker threads for the `HTTP` API service
- Default value: `1`
- Minimum value: `1`

`grpc-compression-type`

- The compression algorithm for gRPC messages
- Available values: `none`, `deflate`, `gzip`
- Default value: `none`

`grpc-concurrency`

- The number of gRPC worker threads
- Default value: `4`
- Minimum value: `1`

`grpc-concurrent-stream`

- The maximum number of concurrent requests allowed in a gRPC stream
- Default value: `1024`
- Minimum value: `1`

`grpc-raft-conn-num`

- The maximum number of links among TiKV nodes for Raft communication
- Default: `1`
- Minimum value: `1`

`grpc-stream-initial-window-size`

- The window size of the gRPC stream
- Default: 2MB
- Unit: KB|MB|GB
- Minimum value: `1KB`

`grpc-keepalive-time`

- The time interval at which that gRPC sends `keepalive` Ping messages
- Default: `10s`
- Minimum value: `1s`

`grpc-keepalive-timeout`

- Disables the timeout for gRPC streams
- Default: `3s`
- Minimum value: `1s`

`concurrent-send-snap-limit`

- The maximum number of snapshots that can be sent at the same time
- Default value: `32`
- Minimum value: `1`

`concurrent-recv-snap-limit`

- The maximum number of snapshots that can be received at the same time
- Default value: `32`
- Minimum value: `1`

`end-point-recursion-limit`

- The maximum number of recursive layers allowed when TiKV decodes the Coprocessor DAG expression
- Default value: `1000`
- Minimum value: `1`

`end-point-slow-log-threshold`

- The time threshold for a TiDB's push down request to print slow log
- Default value: `"1s"`
- Minimum value: `0`

`end-point-request-max-handle-duration`

- The longest duration allowed for a TiDB request to TiKV for processing tasks
- Default value: `60s`
- Minimum value: `1s`

`snap-max-write-bytes-per-sec`

- The maximum allowable disk bandwidth for processing snapshots
- Default value: `1000MB`
- Unit: KB|MB|GB
- Minimum value: `1KB`

4.2.3.2.2 readpool.storage

Configuration items related to storage thread pool

`high-concurrency`

- The allowable number of concurrent threads that handle high-priority `read` requests
- Default value: `4`
- Minimum value: `1`

`normal-concurrency`

- The allowable number of concurrent threads that handle normal-priority `read` requests
- Default value: `4`
- Minimum value: `1`

`low-concurrency`

- The allowable number of concurrent threads that handle low-priority `read` requests
- Default value: `4`
- Minimum value: `1`

`max-tasks-per-worker-high`

- The maximum number of tasks allowed for a single thread in a high-priority thread pool. `Server Is Busy` is returned when the value is exceeded.
- Default value: `2000`
- Minimum value: `2`

`max-tasks-per-worker-normal`

- The maximum number of tasks allowed for a single thread in a normal-priority thread pool. `Server Is Busy` is returned when the value is exceeded.
- Default value: `2000`
- Minimum value: `2`

`max-tasks-per-worker-low`

- The maximum number of tasks allowed for a single thread in a low-priority thread pool. `Server Is Busy` is returned when the value is exceeded.
- Default value: `2000`
- Minimum value: `2`

`stack-size`

- The stack size of threads in the Storage read thread pool
- Default value: `10MB`
- Unit: KB|MB|GB
- Minimum value: `2MB`

4.2.3.2.3 `readpool.coprocessor`

Configuration items related to the Coprocessor thread pool

`high-concurrency`

- The allowable number of concurrent threads that handle high-priority Coprocessor requests, such as checkpoints
- Default value: `CPU * 0.8`
- Minimum value: `1`

`normal-concurrency`

- The allowable number of concurrent threads that handle normal-priority Coprocessor requests
- Default value: `CPU * 0.8`
- Minimum value: `1`

`low-concurrency`

- The allowable number of concurrent threads that handle low-priority Coprocessor requests, such as table scan
- Default value: `CPU * 0.8`
- Minimum value: `1`

`max-tasks-per-worker-high`

- The number of tasks allowed for a single thread in a high-priority thread pool. When this number is exceeded, `Server Is Busy` is returned.
- Default value: `2000`
- Minimum value: `2`

`max-tasks-per-worker-normal`

- The number of tasks allowed for a single thread in a normal-priority thread pool. When this number is exceeded, `Server Is Busy` is returned.
- Default value: `2000`
- Minimum value: `2`

`max-tasks-per-worker-low`

- The number of tasks allowed for a single thread in a low-priority thread pool. When this number is exceeded, `Server Is Busy` is returned.
- Default value: `2000`
- Minimum value: `2`

`stack-size`

- The stack size of the thread in the Coprocessor thread pool
- Default value: `10MB`
- Unit: KB|MB|GB
- Minimum value: `2MB`

4.2.3.2.4 storage

Configuration items related to storage

`scheduler-notify-capacity`

- The maximum number of messages that `scheduler` gets each time
- Default value: `10240`
- Minimum value: `1`

`scheduler-concurrency`

- A built-in memory lock mechanism to prevent simultaneous operations on a key. Each key has a hash in a different slot.
- Default value: `2048000`

- Minimum value: `1`

`scheduler-worker-pool-size`

- The number of `scheduler` threads, mainly used for checking transaction consistency before data writing
- Default value: `4`
- Minimum value: `1`

`scheduler-pending-write-threshold`

- The maximum size of the write queue. A `Server Is Busy` error is returned for a new write to TiKV when this value is exceeded.
- Default value: `100MB`
- Unit: MB|GB

4.2.3.2.5 raftstore

Configuration items related to Raftstore

`sync-log`

- Enables or disables synchronous write mode. In the synchronous write mode, each commit is forced to be flushed to raft-log synchronously for persistent storage.
- Default value: `true`

> **Warning:**
>
> Setting the value to `false` might lead to **data loss**. It is **strongly recommended** that you do not modify this configuration.

`prevote`

- Enables or disables `prevote`. Enabling this feature helps reduce jitter on the system after recovery from network partition.
- Default value: `true`

`raftdb-path`

- The path to the Raft library, which is `storage.data-dir/raft` by default
- Default value: “”

`raft-base-tick-interval`

- The time interval at which the Raft state machine ticks
- Default value: `1s`
- Minimum value: greater than `0`

`raft-heartbeat-ticks`

- The number of passed ticks when the heartbeat is sent. This means that a heartbeat is sent at the time interval of `raft-base-tick-interval` * `raft-heartbeat-ticks`.
- Default value: `2`
- Minimum value: greater than `0`

`raft-election-timeout-ticks`

- The number of passed ticks when Raft election is initiated. This means that if Raft group is missing the leader, a leader election is initiated approximately after the time interval of `raft-base-tick-interval` * `raft-election-timeout-ticks`.
- Default value: `10`
- Minimum value: `raft-heartbeat-ticks`

`raft-min-election-timeout-ticks`

- The minimum number of ticks during which the Raft election is initiated. If the number is `0`, the value of `raft-election-timeout-ticks` is used. The value of this parameter must be greater than or equal to `raft-election-timeout-ticks`.
- Default value: `0`
- Minimum value: `0`

`raft-max-election-timeout-ticks`

- The maximum number of ticks during which the Raft election is initiated. If the number is `0`, the value of `raft-election-timeout-ticks` * `2` is used.
- Default value: `0`
- Minimum value: `0`

`raft-max-size-per-message`

- The soft limit on the size of a single message packet
- Default value: `1MB`
- Minimum value: `0`
- Unit: MB

`raft-max-inflight-msgs`

- The number of Raft logs to be confirmed. If this number is exceeded, log sending slows down.
- Default value: `256`
- Minimum value: greater than `0`

`raft-entry-max-size`

- The hard limit on the maximum size of a single log
- Default value: `8MB`
- Minimum value: `0`
- Unit: MB|GB

`raft-log-gc-tick-interval`

- The time interval at which the polling task of deleting Raft logs is scheduled. `0` means that this feature is disabled.
- Default value: `10s`
- Minimum value: `0`

`raft-log-gc-threshold`

- The soft limit on the maximum allowable count of residual Raft logs
- Default value: `50`
- Minimum value: `1`

`raft-log-gc-count-limit`

- The hard limit on the allowable number of residual Raft logs
- Default value: the log number that can be accommodated in the 3/4 Region size (calculated as 1MB for each log)
- Minimum value: `0`

`raft-log-gc-size-limit`

- The hard limit on the allowable size of residual Raft logs
- Default value: 3/4 of the Region size
- Minimum value: greater than `0`

`raft-entry-cache-life-time`

- The maximum remaining time allowed for the log cache in memory.
- Default value: `30s`
- Minimum value: `0`

`raft-reject-transfer-leader-duration`

- The protection time for new nodes, which is used to control the shortest interval to migrate a leader to the newly added node. Setting this value too small might cause the failure of leader transfer.
- Default value: `3s`
- Minimum value: `0`

`hibernate-regions` (**Experimental**)

- Enables or disables Hibernate Region. When this option is enabled, a Region idle for a long time is automatically set as hibernated. This reduces the extra overhead caused by heartbeat messages between the Raft leader and the followers for idle Regions. You can use `raftstore.peer-stale-state-check-interval` to modify the heartbeat interval between the leader and the followers of hibernated Regions.
- Default value: false

`raftstore.peer-stale-state-check-interval`

- Modifies the state check interval for Regions.
- Default value: 5 min

`split-region-check-tick-interval`

- Specifies the interval at which to check whether the Region split is needed. `0` means that this feature is disabled.
- Default value: `10s`
- Minimum value: `0`

`region-split-check-diff`

- The maximum value by which the Region data is allowed to exceed before Region split
- Default value: 1/16 of the Region size.
- Minimum value: `0`

`region-compact-check-interval`

- The time interval at which to check whether it is necessary to manually trigger RocksDB compaction. `0` means that this feature is disabled.

- Default value: `5m`
- Minimum value: `0`

`clean-stale-peer-delay`

- Delays the time in deleting expired replica data
- Default value: `10m`
- Minimum value: `0`

`region-compact-check-step`

- The number of Regions checked at one time for each round of manual compaction
- Default value: `100`
- Minimum value: `0`

`region-compact-min-tombstones`

- The number of tombstones required to trigger RocksDB compaction
- Default value: `10000`
- Minimum value: `0`

`region-compact-tombstones-percent`

- The proportion of tombstone required to trigger RocksDB compaction
- Default value: `30`
- Minimum value: `1`
- Maximum value: `100`

`pd-heartbeat-tick-interval`

- The time interval at which a Region's heartbeat to PD is triggered. `0` means that this feature is disabled.
- Default value: `1m`
- Minimum value: `0`

`pd-store-heartbeat-tick-interval`

- The time interval at which a store's heartbeat to PD is triggered. `0` means that this feature is disabled.
- Default value: `10s`
- Minimum value: `0`

`snap-mgr-gc-tick-interval`

- The time interval at which the recycle of expired snapshot files is triggered. `0` means that this feature is disabled.
- Default value: `5s`
- Minimum value: `0`

`snap-gc-timeout`

- The longest time for which a snapshot file is saved
- Default value: `4h`
- Minimum value: `0`

`lock-cf-compact-interval`

- The time interval at which TiKV triggers a manual compaction for the Lock Column Family
- Default value: `10m`
- Minimum value: `0`

`lock-cf-compact-bytes-threshold`

- The size out of which TiKV triggers a manual compaction for the Lock Column Family
- Default value: `256MB`
- Minimum value: `0`
- Unit: MB

`notify-capacity`

- The longest length of the Region message queue.
- Default value: `40960`
- Minimum value: `0`

`messages-per-tick`

- The maximum number of messages processed per batch
- Default value: `4096`
- Minimum value: `0`

`max-peer-down-duration`

- The longest inactive duration allowed for a peer. A peer with timeout is marked as `down`, and PD tries to delete it later.

- Default value: `5m`
- Minimum value: `0`

`max-leader-missing-duration`

- The longest duration allowed for a peer to be in the state where a Raft group is missing the leader. If this value is exceeded, the peer verifies with PD whether the peer has been deleted.
- Default value: `2h`
- Minimum value: greater than `abnormal-leader-missing-duration`

`abnormal-leader-missing-duration`

- The longest duration allowed for a peer to be in the state where a Raft group is missing the leader. If this value is exceeded, the peer is seen as abnormal and marked in metrics and logs.
- Default value: `10m`
- Minimum value: greater than `peer-stale-state-check-interval`

`peer-stale-state-check-interval`

- The time interval to trigger the check for whether a peer is in the state where a Raft group is missing the leader.
- Default value: `5m`
- Minimum value: greater than `2 * election-timeout`

`leader-transfer-max-log-lag`

- The maximum number of missing logs allowed for the transferee during a Raft leader transfer
- Default value: `10`
- Minimum value: `10`

`snap-apply-batch-size`

- The memory cache size required when the imported snapshot file is written into the disk
- Default value: `10MB`
- Minimum value: `0`
- Unit: MB

`consistency-check-interval`

- The time interval at which the consistency check is triggered. `0` means that this feature is disabled.
- Default value: `0s`
- Minimum value: `0`

`raft-store-max-leader-lease`

- The longest trusted period of a Raft leader
- Default value: `9s`
- Minimum value: `0`

`allow-remove-leader`

- Determines whether to allow deleting the main switch
- Default value: `false`

`right-derive-when-split`

- Specifies the start key of the new Region when a Region is split. When this configuration item is set to `true`, the start key is the maximum split key. When this configuration item is set to `false`, the start key is the original Region's start key.
- Default value: `true`

`merge-max-log-gap`

- The maximum number of missing logs allowed when `merge` is performed
- Default value: `10`
- Minimum value: greater than `raft-log-gc-count-limit`

`merge-check-tick-interval`

- The time interval at which TiKV checks whether a Region needs merge
- Default value: `10s`
- Minimum value: greater than `0`

`use-delete-range`

> **Warning:**
>
> This is still an experimental feature. It is recommended **not** to use this feature in the production environment.

- Determines whether to delete data from the `rocksdb delete_range` interface
- Default value: `false`

`cleanup-import-sst-interval`

- The time interval at which the expired SST file is checked. `0` means that this feature is disabled.
- Default value: `10m`
- Minimum value: `0`

`local-read-batch-size`

- The maximum number of read requests processed in one batch
- Default value: `1024`
- Minimum value: greater than `0`

`apply-max-batch-size`

- The maximum number of requests for data flushing in one batch
- Default value: `1024`
- Minimum value: greater than `0`

`apply-pool-size`

- The allowable number of threads in the pool that flushes data to storage
- Default value: `2`
- Minimum value: greater than `0`

`store-max-batch-size`

- The maximum number of requests processed in one batch
- Default value: `1024`
- Minimum value: greater than `0`

`store-pool-size`

- The allowable number of threads that process Raft
- Default value: `2`
- Minimum value: greater than `0`

`future-poll-size`

- The allowable number of threads that drive `future`
- Default value: `1`
- Minimum value: greater than `0`

4.2.3.2.6 coprocessor

Configuration items related to Coprocessor

`split-region-on-table`

- Determines whether to split Region by table. It is recommended for you to use the feature only in TiDB mode.
- Default value: `true`

`batch-split-limit`

- The threshold of Region split in batches. Increasing this value speeds up Region split.
- Default value: `10`
- Minimum value: `1`

`region-max-size`

- The maximum size of a Region. When the value is exceeded, the Region splits into many.
- Default value: `144MB`
- Unit: KB|MB|GB

`region-split-size`

- The size of the newly split Region. This value is an estimate.
- Default value: `96MB`
- Unit: KB|MB|GB

`region-max-keys`

- The maximum allowable number of keys in a Region. When this value is exceeded, the Region splits into many.
- Default value: `1440000`

`region-split-keys`

- The number of keys in the newly split Region. This value is an estimate.
- Default value: `960000`

4.2.3.2.7 rocksdb

Configuration items related to RocksDB

`max-background-jobs`

- The number of background threads in RocksDB
- Default value: `8`
- Minimum value: `1`

`max-sub-compactions`

- The number of sub-compaction operations performed concurrently in RocksDB
- Default value: `1`
- Minimum value: `1`

`max-open-files`

- The total number of files that RocksDB can open
- Default value: `40960`
- Minimum value: `-1`

`max-manifest-file-size`

- The maximum size of a RocksDB Manifest file
- Default value: `128MB`
- Minimum value: `0`
- Unit: B|KB|MB|GB

`create-if-missing`

- Determines whether to automatically create a DB switch
- Default value: `true`

`wal-recovery-mode`

- WAL recovery mode
- Available values: `0` (`TolerateCorruptedTailRecords`), `1` (`AbsoluteConsistency`), `2` (`PointInTimeRecovery`), `3` (`SkipAnyCorruptedRecords`)
- Default value: `2`
- Minimum value: `0`
- Maximum value: `3`

`wal-dir`

- The directory in which WAL files are stored
- Default value: `/tmp/tikv/store`

`wal-ttl-seconds`

- The living time of the archived WAL files. When the value is exceeded, the system deletes these files.
- Default value: `0`
- Minimum value: `0`
- unit: second

`wal-size-limit`

- The size limit of the archived WAL files. When the value is exceeded, the system deletes these files.
- Default value: `0`
- Minimum value: `0`
- Unit: B|KB|MB|GB

`enable-statistics`

- Determines whether to enable the statistics of RocksDB
- Default value: `true`

`stats-dump-period`

- Enables or disables Pipelined Write
- Default value: `true`

`compaction-readahead-size`

- The size of `readahead` when compaction is being performed
- Default value: `0`
- Minimum value: `0`
- Unit: B|KB|MB|GB

`writable-file-max-buffer-size`

- The maximum buffer size used in WritableFileWrite
- Default value: `1MB`

- Minimum value: `0`
- Unit: B|KB|MB|GB

`use-direct-io-for-flush-and-compaction`

- Determines whether to use `O_DIRECT` for both reads and writes in background flush and compactions
- Default value: `false`

`rate-bytes-per-sec`

- The maximum rate permitted by Rate Limiter
- Default value: `0`
- Minimum value: `0`
- Unit: Bytes

`rate-limiter-mode`

- Rate LImiter mode
- Available values: `1` (`ReadOnly`), `2` (`WriteOnly`), `3` (`AllIo`)
- Default value: `2`
- Minimum value: `1`
- Maximum value: `3`

`auto-tuned`

- Determines whether to automatically optimize the configuration of the Rate LImiter
- Default value: `false`

`enable-pipelined-write`

- Enables or disables Pipelined Write
- Default value: `true`

`bytes-per-sync`

- The rate at which OS incrementally synchronizes files to disk while these files are being written asynchronously
- Default value: `1MB`
- Minimum value: `0`
- Unit: B|KB|MB|GB

`wal-bytes-per-sync`

- The rate at which OS incrementally synchronizes WAL files to disk while the WAL files are being written
- Default value: `512KB`
- Minimum value: `0`
- Unit: B|KB|MB|GB

`info-log-max-size`

- The maximum size of Info log
- Default value: `1GB`
- Minimum value: `0`
- Unit: B|KB|MB|GB

`info-log-roll-time`

- The time interval at which Info logs are truncated. If the value is `0s`, logs are not truncated.
- Default value: `0s`

`info-log-keep-log-file-num`

- The maximum number of kept log files
- Default value: `10`
- Minimum value: `0`

`info-log-dir`

- The directory in which logs are stored
- Default value: “”

4.2.3.2.8 rocksdb.titan

Configuration items related to Titan

> **Warning:**
>
> This is still an experimental feature. It is recommended **not** to use this feature in the production environment.

`enabled`

- Enables or disables Titan
- Default value: `false`

`dirname`

- The directory in which the Titan Blob file is stored
- Default value: `titandb`

`disable-gc`

- Determines whether to disable Garbage Collection (GC) that Titan performs to Blob files
- Default value: `false`

`max-background-gc`

- The maximum number of GC threads in Titan
- Default value: `1`
- Minimum value: `1`

4.2.3.2.9 rocksdb.defaultcf | rocksdb.writecf | rocksdb.lockcf

Configuration items related to `rocksdb.defaultcf`, `rocksdb.writecf`, and `rocksdb.lockcf`.

`block-size`

- The default size of a RocksDB block
- Default value for `defaultcf` and `writecf`: `"64KB"`
- Default value for `lockcf`: `"16KB"`
- Minimum value: `"1KB"`
- Unit: KB|MB|GB

`block-cache-size`

- The cache size of a RocksDB block
- Default value for `defaultcf`: `Total machine memory * 25%`
- Default value for `writecf`: `Total machine memory * 15%`
- Default value for `lockcf`: `Total machine memory * 2%`
- Minimum value: `0`
- Unit: KB|MB|GB

`disable-block-cache`

- Enables or disables block cache
- Default value: `false`

`cache-index-and-filter-blocks`

- Enables or disables caching index and filter
- Default value: `true`

`pin-l0-filter-and-index-blocks`

- Determines whether to pin the index and filter at L0
- Default value: `true`

`use-bloom-filter`

- Enables or disables bloom filter
- Default value: `true`

`optimize-filters-for-hits`

- Determines whether to optimize the hit ratio of filters
- Default value for `defaultcf`: `true`
- Default value for `writecf` and `lockcf`: `false`

`whole-key-filtering`

- Determines whether to put the entire key to bloom filter
- Default value for `defaultcf`and `lockcf`: `true`
- Default value for `writecf`: `false`

`bloom-filter-bits-per-key`

- The length that bloom filter reserves for each key
- Default value: `10`
- unit: byte

`block-based-bloom-filter`

- Determines whether each block creates a bloom filter
- Default value: `false`

`read-amp-bytes-per-bit`

- Enables or disables statistics of read amplification.
- Available values: `0` (disabled), > `0` (enabled).
- Default value: `0`
- Minimum value: `0`

`compression-per-level`

- The default compression algorithm for each level
- Optional values: ["no", "no", "lz4", "lz4", "lz4", "zstd", "zstd"]
- Default value for `defaultcf` and `writecf`: ["no", "no", "lz4", "lz4", "lz4", "zstd", "zstd"]
- Default value for `lockcf`: ["no", "no", "no", "no", "no", "no", "no"]

`write-buffer-size`

- Memtable size
- Default value: `"128MB"`
- Minimum value: `0`
- Unit: KB|MB|GB

`max-write-buffer-number`

- The maximum number of memtables
- Default value: `5`
- Minimum value: `0`

`min-write-buffer-number-to-merge`

- The minimum number of memtables required to trigger flush
- Default value: `1`
- Minimum value: `0`

`max-bytes-for-level-base`

- The maximum number of bytes at base level (L1). Generally, it is set to 4 times the size of a memtable.
- Default value for `defaultcf` and `writecf`: `"512MB"`
- Default value for `lockcf`: `"128MB"`
- Minimum value: `0`
- Unit: KB|MB|GB

`target-file-size-base`

- The size of the target file at base level
- Default: `8MB`
- Minimum value: `0`
- Unit: KB|MB|GB

`level0-file-num-compaction-trigger`

- The maximum number of files at L0 that trigger compaction
- Default value for `defaultcf` and `writecf`: `4`
- Default value for `lockcf`: `1`
- Minimum value: `0`

`level0-slowdown-writes-trigger`

- The maximum number of files at L0 that trigger write stall
- Default value: `20`
- Minimum value: `0`

`level0-stop-writes-trigger`

- The maximum number of files at L0 required to completely block write
- Default value: `36`
- Minimum value: `0`

`max-compaction-bytes`

- The maximum number of bytes written into disk per compaction
- Default value: `2GB`
- Minimum value: `0`
- Unit: KB|MB|GB

`compaction-pri`

- The priority type of compaction
- Optional values: `0` (`ByCompensatedSize`), `1` (`OldestLargestSeqFirst`), `2` (`OldestSmallestSeqFirs` ↪), `3` (`MinOverlappingRatio`)
- Default value for `defaultcf` and `writecf`: `3`
- Default value for `lockcf`: `0`

`dynamic-level-bytes`

- Determines whether to optimize dynamic level bytes
- Default value: `true`

`num-levels`

- The maximum number of levels in a RocksDB file
- Default value: `7`

`max-bytes-for-level-multiplier`

- The default amplification multiple for each layer
- Default value: `10`

`compaction-style`

- Compaction method
- Available values: `level`, `universal`
- Default value: `level`

`disable-auto-compactions`

- Enables or disables automatic compaction
- Default value: `false`

`soft-pending-compaction-bytes-limit`

- The soft limit on the pending compaction bytes
- Default value: `64GB`
- Unit: KB|MB|GB

`hard-pending-compaction-bytes-limit`

- The hard limit on the pending compaction bytes
- Default value: `256GB`
- Unit: KB|MB|GB

4.2.3.2.10 `rocksdb.defaultcf.titan`

Configuration items related to `rocksdb.defaultcf.titan`.

`min-blob-size`

- The smallest value stored in a Blob file. Values smaller than the specified size are stored in the LSM-Tree.
- Default value: `1KB`
- Minimum value: `0`
- Unit: KB|MB|GB

`blob-file-compression`

- The compression algorithm used in a Blob file
- Available values: `no`, `snappy`, `zlib`, `bzip2`, `lz4`, `lz4hc`, `zstd`
- Default value: `lz4`

`blob-cache-size`

- The cache size of a Blob file
- Default value: `0GB`
- Minimum value: `0`
- Unit: KB|MB|GB

`min-gc-batch-size`

- The minimum total size of Blob files required to perform GC for one time
- Default value: `16MB`
- Minimum value: `0`
- Unit: KB|MB|GB

`max-gc-batch-size`

- The maximum total size of Blob files allowed to perform GC for one time
- Default value: `64MB`
- Minimum value: `0`
- Unit: KB|MB|GB

`discardable-ratio`

- The ratio at which GC is triggered for Blob files. The Blob file can be selected for GC only if the proportion of the invalid values in a Blob file exceeds this ratio.

- Default value: `0.5`
- Minimum value: `0`
- Maximum value: `1`

`sample-ratio`

- The ratio of (data read from a Blob file/the entire Blob file) when sampling the file during GC
- Default value: `0.1`
- Minimum value: `0`
- Maximum value: `1`

`merge-small-file-threshold`

- When the size of a Blob file is smaller than this value, the Blob file might still be selected for GC. In this situation, `discardable-ratio` is ignored.
- Default value: `8MB`
- Minimum value: `0`
- Unit: KB|MB|GB

4.2.3.2.11 `raftdb`

Configuration items related to `raftdb`

`max-background-jobs`

- The number of background threads in RocksDB
- Default value: `2`
- Minimum value: `1`

`max-sub-compactions`

- The number of concurrent sub-compaction operations performed in RocksDB
- Default value: `2`
- Minimum value: `1`

`wal-dir`

- The directory in which WAL files are stored
- Default value: `/tmp/tikv/store`

4.2.3.2.12 security

Configuration items related to security

`ca-path`

- The path of the CA file
- Default value: ""

`cert-path`

- The path of the Privacy Enhanced Mail (PEM) file that contains the X509 certificate
- Default value: ""

`key-path`

- The path of the PEM file that contains the X509 key
- Default value: ""

4.2.3.2.13 import

Configuration items related to `import`

`num-threads`

- The number of threads to process RPC requests
- Default value: `8`
- Minimum value: `1`

`num-import-jobs`

- The number of jobs imported concurrently
- Default value: `8`
- Minimum value: `1`

4.2.3.2.14 pessimistic-txn

`enabled`

- Enables the pessimistic transaction mode. For pessimistic transaction usage, refer to TiDB Pessimistic Transaction Mode.
- Default value: `true`

`wait-for-lock-timeout`

- The max time that a pessimistic transaction in TiKV waits for other transactions to release the lock, in milliseconds. If timed out, an error is returned to TiDB, and TiDB retries to add a lock. The lock wait timeout is set by `innodb_lock_wait_timeout`.
- Default value: 1000
- Minimum value: 1

`wake-up-delay-duration`

- When pessimistic transactions release the lock, among all the transactions waiting for lock, only the transaction with the smallest `start ts` is woken up. Other transactions will be woken up after `wake-up-delay-duration` milliseconds.
- Default value: 20

4.3 Security

4.3.1 Security Compatibility with MySQL

TiDB supports similar security functionality to MySQL 5.7, with the following exceptions:

- Only the `mysql_native_password` authentication scheme is supported
  - In MySQL 8.0, `mysql_native_password` is no longer the default/preferred plugin. To connect to TiDB using a MySQL client from MySQL 8.0, you must explicitly specify `default-auth=mysql_native_password`.
- External authentication (such as with LDAP) is not currently supported
- Column level permissions are not supported
- Password expiry, as well as password last-changed tracking and password lifetime are not supported #9709
- The permission attributes `max_questions`, `max_updated`, `max_connections`, `max_user_connections` are not supported
- Password validation is not currently supported #9741
- Transparent Data Encryption (TDE) is not currently supported

4.3.2 Privilege Management

TiDB supports MySQL 5.7's privilege management system, including the syntax and privilege types. Starting with TiDB 3.0, support for SQL Roles is also available.

This document introduces privilege-related TiDB operations, privileges required for TiDB operations and implementation of the privilege system.

4.3.2.1 Privilege-related operations

4.3.2.1.1 Grant privileges

The `GRANT` statement grants privileges to the user accounts.

For example, use the following statement to grant the `xxx` user the privilege to read the `test` database.

```
GRANT SELECT ON test.* TO 'xxx'@'%';
```

Use the following statement to grant the `xxx` user all privileges on all databases:

```
GRANT ALL PRIVILEGES ON *.* TO 'xxx'@'%';
```

By default, `GRANT` statements will return an error if the user specified does not exist. This behavior depends on if the SQL Mode `NO_AUTO_CREATE_USER` is specified:

```
mysql> SET sql_mode=DEFAULT;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @@sql_mode;
+--
    ↪ ------------------------------------------------------------------------
    ↪
| @@sql_mode
    ↪
    ↪ |
+--
    ↪ ------------------------------------------------------------------------
    ↪
| ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,
    ↪ ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
    ↪  |
+--
    ↪ ------------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)

mysql> SELECT * FROM mysql.user WHERE user='idontexist';
Empty set (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON test.* TO 'idontexist';
ERROR 1105 (HY000): You are not allowed to create a user with GRANT

mysql> SELECT user,host,password FROM mysql.user WHERE user='idontexist';
Empty set (0.00 sec)
```

In the following example, the user `idontexist` is automatically created with an empty password because the SQL Mode `NO_AUTO_CREATE_USER` was not set. This is **not recom-**

mended since it presents a security risk: miss-spelling a username will result in a new user created with an empty password:

```
mysql> SET @@sql_mode='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,
    ↪ NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,
    ↪ NO_ENGINE_SUBSTITUTION';
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @@sql_mode;
+--
    ↪ ----------------------------------------------------------------------------
    ↪
| @@sql_mode
    ↪
    ↪ |
+--
    ↪ ----------------------------------------------------------------------------
    ↪
| ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,
    ↪ ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION |
+--
    ↪ ----------------------------------------------------------------------------
    ↪
1 row in set (0.00 sec)

mysql> SELECT * FROM mysql.user WHERE user='idontexist';
Empty set (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON test.* TO 'idontexist';
Query OK, 1 row affected (0.05 sec)

mysql> SELECT user,host,password FROM mysql.user WHERE user='idontexist';
+------------+------+----------+
| user       | host | password |
+------------+------+----------+
| idontexist | %    |          |
+------------+------+----------+
1 row in set (0.00 sec)
```

You can use fuzzy matching in `GRANT` to grant privileges to databases.

```
mysql> GRANT ALL PRIVILEGES ON `te%`.* TO genius;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT user,host,db FROM mysql.db WHERE user='genius';
+--------/------/-----+
```

```
| user  | host | db |
+--------/------/-----+
| genius | %  | te% |
+--------/------/-----+
1 row in set (0.00 sec)
```

In this example, because of the % in te%, all the databases starting with te are granted the privilege.

4.3.2.1.2 Revoke privileges

The REVOKE statement enables system administrators to revoke privileges from the user accounts.

The REVOKE statement corresponds with the REVOKE statement:

```
REVOKE ALL PRIVILEGES ON `test`.* FROM 'genius'@'localhost';
```

> **Note:**
>
> To revoke privileges, you need the exact match. If the matching result cannot be found, an error will be displayed:

```
mysql> REVOKE ALL PRIVILEGES ON `te%`.* FROM 'genius'@'%';
ERROR 1141 (42000): There is no such grant defined for user 'genius' on
    ↪ host '%'
```

About fuzzy matching, escape, string and identifier:

```
mysql> GRANT ALL PRIVILEGES ON `te\%`.* TO 'genius'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

This example uses exact match to find the database named te%. Note that the % uses the \ escape character so that % is not considered as a wildcard.

A string is enclosed in single quotation marks(”), while an identifier is enclosed in backticks (“). See the differences below:

```
mysql> GRANT ALL PRIVILEGES ON 'test'.* TO 'genius'@'localhost';
ERROR 1064 (42000): You have an error in your SQL syntax; check the
manual that corresponds to your MySQL server version for the right
syntax to use near ''test'.* to 'genius'@'localhost'' at line 1

mysql> GRANT ALL PRIVILEGES ON `test`.* TO 'genius'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

If you want to use special keywords as table names, enclose them in backticks (“). For example:

```
mysql> CREATE TABLE `select` (id int);
Query OK, 0 rows affected (0.27 sec)
```

4.3.2.1.3 Check privileges granted to users

You can use the `SHOW GRANTS` statement to see what privileges are granted to a user. For example:

```
SHOW GRANTS; -- show grants for the current user

+-------------------------------------------------------------+
| Grants for User                                             |
+-------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' WITH GRANT OPTION |
+-------------------------------------------------------------+
SHOW GRANTS FOR 'root'@'%'; -- show grants for a specific user
```

For example, create a user `rw_user@192.168.%` and grant the user with write privilege on the `test.write_table` table and global read privilege.

```
CREATE USER `rw_user`@`192.168.%`;
GRANT SELECT ON *.* TO `rw_user`@`192.168.%`;
GRANT INSERT, UPDATE ON `test`.`write_table` TO `rw_user`@`192.168.%`;
```

Show granted privileges of the `rw_user@192.168.%` user:

```
SHOW GRANTS FOR `rw_user`@`192.168.%`;

+-----------------------------------------------------------------+
| Grants for rw_user@192.168.%                                    |
+-----------------------------------------------------------------+
| GRANT Select ON *.* TO 'rw_user'@'192.168.%'                    |
| GRANT Insert,Update ON test.write_table TO 'rw_user'@'192.168.%' |
+-----------------------------------------------------------------+
```

4.3.2.2 Privileges required for TiDB operations

You can check privileges of TiDB users in the `INFORMATION_SCHEMA.USER_PRIVILEGES` table.

Privilege type	Privilege variable	Privilege description
ALL	`AllPriv`	All the privileges
Drop	`DropPriv`	Deletes a schema or table

Privilege type	Privilege variable	Privilege description
Index	`IndexPriv`	Creates or deletes an index
Alter	`AlterPriv`	Executes the `ALTER` statement
Super	`SuperPriv`	All the privileges
Create	`CreatePriv`	Creates a schema or table
Select	`SelectPriv`	Reads the table data
Insert	`InsertPriv`	Inserts data to a table
Update	`UpdatePriv`	Updates the table data
Delete	`DeletePriv`	Deleted the table data
Reload	`ReloadPriv`	Executes the `FLUSH` statement
Config	`ConfigPriv`	Dynamically reloads configuration
Trigger	`TriggerPriv`	/
Process	`ProcessPriv`	Displays the running task
Execute	`ExecutePriv`	Executes the `EXECUTE` statement
Drop Role	`DropRolePriv`	Executes `DROP ROLE`
Show View	`ShowViewPriv`	Executes `SHOW CREATE VIEW`
References	`ReferencesPriv`	/
Create View	`CreateViewPriv`	Creates a View
Create User	`CreateUserPriv`	Creates a user
Create Role	`CreateRolePriv`	Executes `CREATE ROLE`
Show Databases	`ShowDBPriv`	Shows the table status in the database

4.3.2.2.1 ALTER

- For all `ALTER` statements, users must have the `ALTER` privilege for the corresponding table.
- For statements except `ALTER...DROP` and `ALTER...RENAME TO`, users must have the `INSERT` and `CREATE` privileges for the corresponding table.
- For the `ALTER...DROP` statement, users must have the `DROP` privilege for the corresponding table.
- For the `ALTER...RENAME TO` statement, users must have the `DROP` privilege for the table before renaming, and the `CREATE` and `INSERT` privileges for the table after renaming.

Note:

In MySQL 5.7 documentation, users need `INSERT` and `CREATE` privileges to perform the `ALTER` operation on a table. But in reality for MySQL 5.7.25, only the `ALTER` privilege is required in this case. Currently, the `ALTER` privilege in TiDB is consistent with the actual behavior in MySQL.

4.3.2.2.2 CREATE DATABASE

Requires the `CREATE` privilege for the database.

4.3.2.2.3 CREATE INDEX

Requires the `INDEX` privilege for the table.

4.3.2.2.4 CREATE TABLE

Requires the `CREATE` privilege for the table.

To execute the `CREATE TABLE...LIKE...` statement, the `SELECT` privilege for the table is required.

4.3.2.2.5 CREATE VIEW

Requires the `CREATE VIEW` privilege.

> **Note:**
>
> If the current user is not the user that creates the View, both the `CREATE VIEW` and `SUPER` privileges are required.

4.3.2.2.6 DROP DATABASE

Requires the `DROP` privilege for the table.

4.3.2.2.7 DROP INDEX

Requires the `INDEX` privilege for the table.

4.3.2.2.8 DROP TABLES

Requires the `DROP` privilege for the table.

4.3.2.2.9 LOAD DATA

Requires the `INSERT` privilege for the table.

4.3.2.2.10 TRUNCATE TABLE

Requires the `DROP` privilege for the table.

4.3.2.2.11 RENAME TABLE

Requires the `ALTER` and `DROP` privileges for the table before renaming and the `CREATE` and `INSERT` privileges for the table after renaming.

4.3.2.2.12 ANALYZE TABLE

Requires the `INSERT` and `SELECT` privileges for the table.

4.3.2.2.13 SHOW

`SHOW CREATE TABLE` requires any single privilege to the table.

`SHOW CREATE VIEW` requires the `SHOW VIEW` privilege.

`SHOW GRANTS` requires the `SELECT` privilege to the `mysql` database. If the target user is current user, `SHOW GRANTS` does not require any privilege.

4.3.2.2.14 CREATE ROLE/USER

`CREATE ROLE` requires the `CREATE ROLE` privilege.

`CREATE USER` requires the `CREATE USER` privilege.

4.3.2.2.15 DROP ROLE/USER

`DROP ROLE` requires the `DROP ROLE` privilege.

`DROP USER` requires the `CREATE USER` privilege.

4.3.2.2.16 ALTER USER

Requires the `CREATE USER` privilege.

4.3.2.2.17 GRANT

Requires the `GRANT` privilege with the privileges granted by `GRANT`.

Requires additional `CREATE USER` privilege to create a user implicitly.

`GRANT ROLE` requires `SUPER` privilege.

4.3.2.2.18 REVOKE

Requires the `GRANT` privilege and those privileges targeted by the `REVOKE` statement.

`REVOKE ROLE` requires `SUPER` privilege.

4.3.2.2.19 SET GLOBAL

Requires `SUPER` privilege to set global variables.

4.3.2.2.20 ADMIN

Requires `SUPER` privilege.

4.3.2.2.21 SET DEFAULT ROLE

Requires `SUPER` privilege.

4.3.2.2.22 KILL

Requires `SUPER` privilege to kill other user sessions.

4.3.2.3 Implementation of the privilege system

4.3.2.3.1 Privilege table

The following system tables are special because all the privilege-related data is stored in them:

- `mysql.user` (user account, global privilege)
- `mysql.db` (database-level privilege)
- `mysql.tables_priv` (table-level privilege)
- `mysql.columns_priv` (column-level privilege; not currently supported)

These tables contain the effective range and privilege information of the data. For example, in the `mysql.user` table:

```
mysql> SELECT User,Host,Select_priv,Insert_priv FROM mysql.user LIMIT 1;
+------/------/-------------/-------------+
| User | Host | Select_priv | Insert_priv |
+------/------/-------------/-------------+
| root | %  | Y           | Y           |
+------/------/-------------/-------------+
1 row in set (0.00 sec)
```

In this record, `Host` and `User` determine that the connection request sent by the `root` user from any host (`%`) can be accepted. `Select_priv` and `Insert_priv` mean that the user has global `Select` and `Insert` privilege. The effective range in the `mysql.user` table is global.

`Host` and `User` in `mysql.db` determine which databases users can access. The effective range is the database.

Note:

It is recommended to only update the privilege tables via the supplied syntax such as `GRANT`, `CREATE USER` and `DROP USER`. Making direct edits to the underlying privilege tables will not automatically update the privilege cache, leading to unpredictable behavior until `FLUSH PRIVILEGES` is executed.

4.3.2.3.2 Connection verification

When the client sends a connection request, TiDB server will verify the login operation. TiDB server first checks the `mysql.user` table. If a record of `User` and `Host` matches the connection request, TiDB server then verifies the `Password`.

User identity is based on two pieces of information: `Host`, the host that initiates the connection, and `User`, the user name. If the user name is not empty, the exact match of user named is a must.

`User`+`Host` may match several rows in `user` table. To deal with this scenario, the rows in the `user` table are sorted. The table rows will be checked one by one when the client connects; the first matched row will be used to verify. When sorting, Host is ranked before User.

4.3.2.3.3 Request verification

When the connection is successful, the request verification process checks whether the operation has the privilege.

For database-related requests (`INSERT`, `UPDATE`), the request verification process first checks the user's global privileges in the `mysql.user` table. If the privilege is granted, you can access directly. If not, check the `mysql.db` table.

The `user` table has global privileges regardless of the default database. For example, the `DELETE` privilege in `user` can apply to any row, table, or database.

In the `db` table, an empty user is to match the anonymous user name. Wildcards are not allowed in the `User` column. The value for the `Host` and `Db` columns can use `%` and `_`, which can use pattern matching.

Data in the `user` and `db` tables is also sorted when loaded into memory.

The use of `%` in `tables_priv` and `columns_priv` is similar, but column value in `Db`, `Table_name` and `Column_name` cannot contain `%`. The sorting is also similar when loaded.

4.3.2.3.4 Time of effect

When TiDB starts, some privilege-check tables are loaded into memory, and then the cached data is used to verify the privileges. Executing privilege management statements such as `GRANT`, `REVOKE`, `CREATE USER`, `DROP USER` will take effect immediately.

Manually editing tables such as `mysql.user` with statements such as `INSERT`, `DELETE`, `UPDATE` will not take effect immediately. This behavior is compatible with MySQL, and privilege cache can be updated with the following statement:

```
FLUSH PRIVILEGES;
```

4.3.3 TiDB User Account Management

This document describes how to manage a TiDB user account.

4.3.3.1 User names and passwords

TiDB stores the user accounts in the table of the `mysql.user` system database. Each account is identified by a user name and the client host. Each account may have a password.

You can connect to the TiDB server using the MySQL client, and use the specified account and password to login:

```
shell> mysql --port 4000 --user xxx --password
```

Or use the abbreviation of command line parameters:

```
shell> mysql -P 4000 -u xxx -p
```

Note:

- To connect to TiDB using a MySQL client from MySQL 8.0, you must explicitly specify `--default-auth=mysql_native_password`, because `mysql_native_password` is no longer the default plugin.

4.3.3.2 Add user accounts

You can create TiDB accounts in two ways:

- By using the standard account-management SQL statements intended for creating accounts and establishing their privileges, such as `CREATE USER` and `GRANT`.
- By manipulating the privilege tables directly with statements such as `INSERT`, `UPDATE`, or `DELETE`.

It is recommended to use the account-management statements, because manipulating the privilege tables directly can lead to incomplete updates. You can also create accounts by using third party GUI tools.

The following example uses the `CREATE USER` and `GRANT` statements to set up four accounts:

```
mysql> CREATE USER 'finley'@'localhost' IDENTIFIED BY 'some_pass';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'finley'@'localhost' WITH GRANT
    ↪ OPTION;
mysql> CREATE USER 'finley'@'%' IDENTIFIED BY 'some_pass';
mysql> GRANT ALL PRIVILEGES ON *.* TO 'finley'@'%' WITH GRANT OPTION;
mysql> CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin_pass';
mysql> GRANT RELOAD,PROCESS ON *.* TO 'admin'@'localhost';
mysql> CREATE USER 'dummy'@'localhost';
```

To see the privileges for an account, use `SHOW GRANTS`:

```
mysql> SHOW GRANTS FOR 'admin'@'localhost';
+-----------------------------------------------------+
| Grants for admin@localhost                          |
+-----------------------------------------------------+
| GRANT RELOAD, PROCESS ON *.* TO 'admin'@'localhost' |
+-----------------------------------------------------+
```

4.3.3.3 Remove user accounts

To remove a user account, use the `DROP USER` statement:

```
mysql> DROP USER 'test'@'localhost';
```

4.3.3.4 Reserved user accounts

TiDB creates the `'root'@'%'` default account during the database initialization.

4.3.3.5 Set account resource limits

Currently, TiDB does not support setting account resource limits.

4.3.3.6 Assign account passwords

TiDB stores passwords in the `mysql.user` system database. Operations that assign or update passwords are permitted only to users with the `CREATE USER` privilege, or, alternatively, privileges for the `mysql` database (`INSERT` privilege to create new accounts, `UPDATE` privilege to update existing accounts).

- To assign a password when you create a new account, use `CREATE USER` and include an `IDENTIFIED BY` clause:

  ```
  CREATE USER 'test'@'localhost' IDENTIFIED BY 'mypass';
  ```

- To assign or change a password for an existing account, use `SET PASSWORD FOR` or `ALTER USER`:

```
SET PASSWORD FOR 'root'@'%' = 'xxx';
```

Or:

```
ALTER USER 'test'@'localhost' IDENTIFIED BY 'mypass';
```

4.3.3.7 Forget the `root` password

1. Modify the configuration file by adding `skip-grant-table` in the `security` part:

```
[security]
skip-grant-table = true
```

2. Use `root` to log in and then modify the password:

```
mysql -h 127.0.0.1 -P 4000 -u root
```

4.3.3.8 `FLUSH PRIVILEGES`

If you modified the privilege tables directly, run the following command to apply changes immediately:

```
FLUSH PRIVILEGES;
```

For details, see Privilege Management.

4.3.4 Role-Based Access Control

The implementation of TiDB's role-based access control (RBAC) system is similar to that of MySQL 8.0. TiDB is compatible with most RBAC syntax of MySQL.

This document introduces TiDB RBAC-related operations and implementation.

4.3.4.1 RBAC operations

A role is a collection of a series of privileges. You can do the following operations:

- Create a role.
- Delete a role.
- Grant a privilege to a role.
- Grant a role to another user. That user can obtain the privileges involved in the role, after enabling the role.

4.3.4.1.1 Create a role

For example, you can use the following statement to create the roles `app_developer`, `app_read`, and `app_write`:

```
CREATE ROLE 'app_developer', 'app_read', 'app_write';
```

For the role naming format and rule, see TiDB User Account Management.

Roles are stored in the `mysql.user` table and the host name part of the role name (if omitted) defaults to `'%'`. The name of the role you are trying to create must be unique; otherwise, an error is reported.

To create a role, you need the `CREATE ROLE` or `CREATE USER` privilege.

4.3.4.1.2 Grant a privilege to a role

The operation of granting a privilege to a role is the same with that of granting a privilege to a user. For details, see TiDB Privilege Management.

For example, you can use the following statement to grant the `app_read` role the privilege to read the `app_db` database:

```
GRANT SELECT ON app_db.* TO 'app_read'@'%';
```

You can use the following statement to grant the `app_write` role the privilege to write data to the `app_db` database:

```
GRANT INSERT, UPDATE, DELETE ON app_db.* TO 'app_write'@'%';;
```

You can use the following statement to grant the `app_developer` role all privileges on the `app_db` database:

```
GRANT ALL ON app_db.* TO 'app_developer';
```

4.3.4.1.3 Grant a role to a user

Assume that a user `dev1` has the developer role with all the privileges on `app_db`; two users `read_user1` and `read_user2` have the read-only privilege on `app_db`; and a user `rw_user1` has read and write privileges on `app_db`.

Use `CREATE USER` to create the users:

```
CREATE USER 'dev1'@'localhost' IDENTIFIED BY 'dev1pass';
CREATE USER 'read_user1'@'localhost' IDENTIFIED BY 'read_user1pass';
CREATE USER 'read_user2'@'localhost' IDENTIFIED BY 'read_user2pass';
CREATE USER 'rw_user1'@'localhost' IDENTIFIED BY 'rw_user1pass';
```

Then use `GRANT` to grant roles to users

```
GRANT 'app_developer' TO 'dev1'@'localhost';
GRANT 'app_read' TO 'read_user1'@'localhost', 'read_user2'@'localhost';
GRANT 'app_read', 'app_write' TO 'rw_user1'@'localhost';
```

To grant a role to another user or revoke a role, you need the `SUPER` privilege.

Granting a role to a user does not mean enabling the role immediately. Enabling a role is another operation.

The following operations might form a "relation loop:"

```
CREATE USER 'u1', 'u2';
CREATE ROLE 'r1', 'r2';

GRANT 'u1' TO 'u1';
GRANT 'r1' TO 'r1';

GRANT 'r2' TO 'u2';
GRANT 'u2' TO 'r2';
```

TiDB supports this multi-level authorization relationship. You can use it to implement privilege inheritance.

4.3.4.1.4 Check a role's privileges

You can use the `SHOW GRANTS` statement to check what privileges have been granted to the user.

To check privilege-related information of another user, you need the `SELECT` privilege on the `mysql` database.

```
SHOW GRANTS FOR 'dev1'@'localhost';
```

```
+-------------------------------------------------+
| Grants for dev1@localhost                       |
+-------------------------------------------------+
| GRANT USAGE ON *.* TO `dev1`@`localhost`        |
| GRANT `app_developer`@`%` TO `dev1`@`localhost` |
+-------------------------------------------------+
```

You can use the `USING` option in `SHOW GRANTS` to check a role's privileges:

```
SHOW GRANTS FOR 'dev1'@'localhost' USING 'app_developer';
```

```
+-------------------------------------------------------------+
| Grants for dev1@localhost                                   |
+-------------------------------------------------------------+
```

```
| GRANT USAGE ON *.* TO `dev1`@`localhost`          |
| GRANT ALL PRIVILEGES ON `app_db`.* TO `dev1`@`localhost` |
| GRANT `app_developer`@`%` TO `dev1`@`localhost`   |
+---------------------------------------------------------+
```

```
SHOW GRANTS FOR 'rw_user1'@'localhost' USING 'app_read', 'app_write';
```

```
+-----------------------------------------------------------------------------+
   ↪
| Grants for rw_user1@localhost                                               |
+-----------------------------------------------------------------------------+
   ↪
| GRANT USAGE ON *.* TO `rw_user1`@`localhost`                                |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `app_db`.* TO `rw_user1`@`
   ↪ localhost` |
| GRANT `app_read`@`%`,`app_write`@`%` TO `rw_user1`@`localhost`              |
+-----------------------------------------------------------------------------+
   ↪
```

```
SHOW GRANTS FOR 'read_user1'@'localhost' USING 'app_read';
```

```
+-----------------------------------------------------+
| Grants for read_user1@localhost                     |
+-----------------------------------------------------+
| GRANT USAGE ON *.* TO `read_user1`@`localhost`      |
| GRANT SELECT ON `app_db`.* TO `read_user1`@`localhost` |
| GRANT `app_read`@`%` TO `read_user1`@`localhost`    |
+-----------------------------------------------------+
```

You can use `SHOW GRANTS` or `SHOW GRANTS FOR CURRENT_USER()` to check the current user's privileges. `SHOW GRANTS` and `SHOW GRANTS FOR CURRENT_USER()` are different in the following aspects:

- `SHOW GRANTS` shows the privilege of the enabled role for the current user.
- `SHOW GRANTS FOR CURRENT_USER()` does not show the enabled role's privilege.

4.3.4.1.5 Set the default role

After a role is granted to a user, it does not take effect immediately. Only after the user enables this role, he can use the privilege the role owns.

You can set default roles for a user. When the user logs in, the default roles are automatically enabled.

```
SET DEFAULT ROLE
    {NONE | ALL | role [, role ] ...}
    TO user [, user ]
```

For example, you can use the following statement to set default roles of `rw_user1@localhost` to `app_read` and `app_write`:

```
SET DEFAULT ROLE app_read, app_write TO 'rw_user1'@'localhost';
```

You can use the following statement to set default roles of `dev1@localhost` to all roles:

```
SET DEFAULT ROLE ALL TO 'dev1'@'localhost';
```

You can use the following statement to disable all default roles of `dev1@localhost`:

```
SET DEFAULT ROLE NONE TO 'dev1'@'localhost';
```

> **Note:**
>
> You need to grant the role to the user before you set the default role to this role.

4.3.4.1.6 Enable a role in the current session

You can enable some role(s) in the current session.

```
SET ROLE {
    DEFAULT
  | NONE
  | ALL
  | ALL EXCEPT role [, role ] ...
  | role [, role ] ...
}
```

For example, after `rw_user1` logs in, you can use the following statement to enable roles `app_read` and `app_write` that are valid only in the current session:

```
SET ROLE 'app_read', 'app_write';
```

You can use the following statement to enable the default role of the current user:

```
SET ROLE DEFAULT
```

You can use the following statement to enable all roles granted to the current user:

```
SET ROLE ALL
```

You can use the following statement to disable all roles:

```
SET ROLE NONE
```

You can use the following statement to enable roles except `app_read`:

```
SET ROLE ALL EXCEPT 'app_read'
```

> **Note:**
>
> If you use `SET ROLE` to enable a role, this role is valid only in the current session.

4.3.4.1.7 Check the current enabled role

The current user can use the `CURRENT_ROLE()` function to check which role has been enabled by the current user.

For example, you can grant default roles to `rw_user1'@'localhost`:

```
SET DEFAULT ROLE ALL TO 'rw_user1'@'localhost';
```

After `rw_user1@localhost` logs in, you can execute the following statement:

```
SELECT CURRENT_ROLE();
```

```
+--------------------------------+
| CURRENT_ROLE()                 |
+--------------------------------+
| `app_read`@`%`,`app_write`@`%` |
+--------------------------------+
```

```
SET ROLE 'app_read'; SELECT CURRENT_ROLE();
```

```
+----------------+
| CURRENT_ROLE() |
+----------------+
| `app_read`@`%` |
+----------------+
```

4.3.4.1.8 Revoke a role

You can use the following statement to revoke the `app_read` role granted to the users `read_user1@localhost` and `read_user2@localhost`:

```
REVOKE 'app_read' FROM 'read_user1'@'localhost', 'read_user2'@'localhost';
```

You can use the following statement to revoke the roles `app_read` and `app_write` granted to the `rw_user1@localhost` user:

```
REVOKE 'app_read', 'app_write' FROM 'rw_user1'@'localhost';
```

The operation of revoking a role from a user is atomic. If you fail to revoke a role, this operation rolls back.

4.3.4.1.9 Revoke a privilege

The `REVOKE` statement is reverse to `GRANT`. You can use `REVOKE` to revoke the privileges of `app_write`.

```
REVOKE INSERT, UPDATE, DELETE ON app_db.* FROM 'app_write';
```

For details, see TiDB Privilege Management.

4.3.4.1.10 Delete a role

You can use the following statement to delete roles `app_read` and `app_write`:

```
DROP ROLE 'app_read', 'app_write';
```

This operation deletes the role records of `app_read` and `app_write` in the `mysql.user` table and related records in the authorization table, and terminates the authorization related to the two roles.

To delete a role, you need the `DROP ROLE` or `DROP USER` privilege.

4.3.4.1.11 Authorization table

In addition to four system privilege tables, the RBAC system introduces two new system privilege tables:

- `mysql.role_edges`: records the authorization relationship of the role and user.
- `mysql.default_roles`: records default roles of each user.

`mysql.role_edges`

`mysql.role_edges` contains the following data:

```
SELECT * FROM mysql.role_edges;
```

```
+-----------+-----------+---------+---------+-------------------+
| FROM_HOST | FROM_USER | TO_HOST | TO_USER | WITH_ADMIN_OPTION |
+-----------+-----------+---------+---------+-------------------+
| %         | r_1       | %       | u_1     | N                 |
+-----------+-----------+---------+---------+-------------------+
1 row in set (0.00 sec)
```

- `FROM_HOST` and `FROM_USER` indicate the role's host name and user name respectively.
- `TO_HOST` and `TO_USER` indicate the host name and user name of the user to which a role is granted.

`mysql.default_roles`

`mysql.default_roles` shows which roles have been enabled by default for each user.

```
SELECT * FROM mysql.default_roles;
```

```
+------+------+-------------------+-------------------+
| HOST | USER | DEFAULT_ROLE_HOST | DEFAULT_ROLE_USER |
+------+------+-------------------+-------------------+
| %    | u_1  | %                 | r_1               |
| %    | u_1  | %                 | r_2               |
+------+------+-------------------+-------------------+
2 rows in set (0.00 sec)
```

- `HOST` and `USER` indicate the user's host name and user name respectively.
- `DEFAULT_ROLE_HOST` and `DEFAULT_ROLE_USER` indicate the host name and user name of the default role respectively.

4.3.4.1.12 References

Because RBAC, user management, and privilege management are closely related, you can refer to operation details in the following resources:

- TiDB Privilege Management
- TiDB User Account Management

4.3.5 Certificate-Based Authentication for Login New in v3.0.8

Starting from v3.0.8, TiDB supports a certificate-based authentication method for users to log into TiDB. With this method, TiDB issues certificates to different users, uses encrypted connections to transfer data, and verifies certificates when users log in. This approach is

more secure than the traditional password-based authentication method commonly used by MySQL users and is thus adopted by an increasing number of users.

To use certificate-based authentication, you might need to perform the following operations:

- Create security keys and certificates
- Configure certificates for TiDB and the client
- Configure the user certificate information to be verified when the user logs in
- Update and replace certificates

The rest of the document introduces in detail how to perform these operations.

4.3.5.1 Create security keys and certificates

4.3.5.1.1 Install OpenSSL

It is recommended that you use OpenSSL to create keys and certificates. Taking the Debian operation system as an example, execute the following command to install OpenSSL:

```
sudo apt-get install openssl
```

4.3.5.1.2 Generate CA key and certificate

1. Execute the following command to generate the CA key:

   ```
   sudo openssl genrsa 2048 > ca-key.pem
   ```

   The output of the above command:

   ```
   Generating RSA private key, 2048 bit long modulus (2 primes)
   ..................+++++
   ..............................................+++++
   e is 65537 (0x010001)
   ```

2. Execute the following command to generate the certificate corresponding to the CA key:

   ```
   sudo openssl req -new -x509 -nodes -days 365000 -key ca-key.pem -out ca
       ↪ -cert.pem
   ```

3. Enter detailed certificate information. For example:

```
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (e.g. city) []:San Francisco
Organization Name (e.g. company) [Internet Widgits Pty Ltd]:PingCAP Inc
    ↪ .
Organizational Unit Name (e.g. section) []:TiDB
Common Name (e.g. server FQDN or YOUR name) []:TiDB admin
Email Address []:s@pingcap.com
```

Note:

In the above certificate details, texts after : are the entered information.

4.3.5.1.3 Generate server key and certificate

1. Execute the following command to generate the server key:

```
sudo openssl req -newkey rsa:2048 -days 365000 -nodes -keyout server-
    ↪ key.pem -out server-req.pem
```

2. Enter detailed certificate information. For example:

```
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (e.g. city) []:San Francisco
Organization Name (e.g. company) [Internet Widgits Pty Ltd]:PingCAP Inc
    ↪ .
Organizational Unit Name (e.g. section) []:TiKV
Common Name (e.g. server FQDN or YOUR name) []:TiKV Test Server
Email Address []:k@pingcap.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. Execute the following command to generate the RSA key of the server:

```
sudo openssl rsa -in server-key.pem -out server-key.pem
```

The output of the above command:

```
writing RSA key
```

4. Use the CA certificate signature to generate the signed server certificate:

```
sudo openssl x509 -req -in server-req.pem -days 365000 -CA ca-cert.pem
    ↪ -CAkey ca-key.pem -set_serial 01 -out server-cert.pem
```

The output of the above command (for example):

```
Signature ok
subject=C = US, ST = California, L = San Francisco, O = PingCAP Inc.,
    ↪ OU = TiKV, CN = TiKV Test Server, emailAddress = k@pingcap.com
Getting CA Private Key
```

> **Note:**
> When you log in, TiDB checks whether the information in the `subject` section of the above output is consistent or not.

4.3.5.1.4 Generate client key and certificate

After generating the server key and certificate, you need to generate the key and certificate for the client. It is often necessary to generate different keys and certificates for different users.

1. Execute the following command to generate the client key:

```
sudo openssl req -newkey rsa:2048 -days 365000 -nodes -keyout client-
    ↪ key.pem -out client-req.pem
```

2. Enter detailed certificate information. For example:

```
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (e.g. city) []:San Francisco
Organization Name (e.g. company) [Internet Widgits Pty Ltd]:PingCAP Inc
    ↪ .
Organizational Unit Name (e.g. section) []:TiDB
Common Name (e.g. server FQDN or YOUR name) []:tpch-user1
Email Address []:zz@pingcap.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. Execute the following command to generate the RSA key of the client:

```
sudo openssl rsa -in client-key.pem -out client-key.pem
```

The output of the above command:

```
writing RSA key
```

4. Use the CA certificate signature to generate the client certificate:

```
sudo openssl x509 -req -in client-req.pem -days 365000 -CA ca-cert.pem
    ↪ -CAkey ca-key.pem -set_serial 01 -out client-cert.pem
```

The output of the above command (for example):

```
Signature ok
subject=C = US, ST = California, L = San Francisco, O = PingCAP Inc.,
    ↪ OU = TiDB, CN = tpch-user1, emailAddress = zz@pingcap.com
Getting CA Private Key
```

> **Note:**
> The information of the `subject` section in the above output is used for certificate configuration for login verification in the `require` section.

4.3.5.1.5 Verify certificate

Execute the following command to verify certificate:

```
openssl verify -CAfile ca-cert.pem server-cert.pem client-cert.pem
```

If the certificate is verified, you will see the following result:

```
server-cert.pem: OK
client-cert.pem: OK
```

4.3.5.2 Configure TiDB and the client to use certificates

After generating the certificates, you need to configure the TiDB server and the client to use the corresponding server certificate or client certificate.

4.3.5.2.1 Configure TiDB to use server certificate

Modify the `[security]` section in the TiDB configuration file. This step specifies the directory in which the CA certificate, the server key, and the server certificate are stored. You can replace `path/to/server-cert.pem`, `path/to/server-key.pem`, `path/to/ca-cert.pem` with your own directory.

```
[security]
ssl-cert ="path/to/server-cert.pem"
ssl-key ="path/to/server-key.pem"
ssl-ca="path/to/ca-cert.pem"
```

Start TiDB and check logs. If the following information is displayed in the log, the configuration is successful:

```
[INFO] [server.go:264] ["secure connection is enabled"] ["client
    ↪ verification enabled"=true]
```

4.3.5.2.2 Configure the client to use client certificate

Configure the client so that the client uses the client key and certificate for login.

Taking the MySQL client as an example, you can use the newly created client certificate, client key and CA by specifying `ssl-cert`, `ssl-key`, and `ssl-ca`:

```
mysql -utest -h0.0.0.0 -P4000 --ssl-cert /path/to/client-cert.new.pem --ssl-
    ↪ key /path/to/client-key.new.pem --ssl-ca /path/to/ca-cert.pem
```

> **Note:**
>
> `/path/to/client-cert.new.pem`, `/path/to/client-key.new.pem`, and `/path/to/ca-cert.pem` are the directory of the CA certificate, client key, and client certificate. You can replace them with your own directory.

4.3.5.3 Configure the user certificate information for login verification

First, connect TiDB using the client to configure the login verification. Then, you can get and configure the user certificate information to be verified.

4.3.5.3.1 Get user certificate information

The user certificate information can be specified by `require subject`, `require issuer`, and `require cipher`, which are used to check the X509 certificate attributes.

- `require subject`: Specifies the `subject` information of the client certificate when you log in. With this option specified, you do not need to configure `require ssl` or x509. The information to be specified is consistent with the entered `subject` information in Generate client keys and certificates.

  To get this option, execute the following command:

```
openssl x509 -noout -subject -in client-cert.pem | sed 's/.\{8\}//' |
    ↪ sed 's/, /\//g' | sed 's/ = /=/g' | sed 's/^/\//'
```

- `require issuer`: Specifies the `subject` information of the CA certificate that issues the user certificate. The information to be specified is consistent with the entered `subject` information in Generate CA key and certificate.

  To get this option, execute the following command:

```
openssl x509 -noout -subject -in ca-cert.pem | sed 's/.\{8\}//' | sed '
    ↪ s/, /\//g' | sed 's/ = /=/g' | sed 's/^/\//'
```

- `require cipher`: Checks the cipher method supported by the client. Use the following statement to check the list of supported cipher methods:

```
SHOW SESSION STATUS LIKE 'Ssl_cipher_list';
```

4.3.5.3.2 Configure user certificate information

After getting the user certificate information (`require subject`, `require issuer`, `require cipher`), configure these information to be verified when creating a user, granting privileges, or altering a user. Replace `<replaceable>` with the corresponding information in the following statements.

You can configure one option or multiple options using the space or `and` as the separator.

- Configure user certificate when creating a user (`create user`):

```
create user 'u1'@'%' require issuer '<replaceable>' subject '<
    ↪ replaceable>' cipher '<replaceable>';
```

- Configure user certificate when granting privileges:

```
grant all on *.* to 'u1'@'%' require issuer '<replaceable>' subject '<
    ↪ replaceable>' cipher '<replaceable>';
```

- Configure user certificate when altering a user:

```
alter user 'u1'@'%' require issuer '<replaceable>' subject '<
    ↪ replaceable>' cipher '<replaceable>';
```

After the above configuration, the following items will be verified when you log in:

- SSL is used; the CA that issues the client certificate is consistent with the CA configured in the server.

- The `issuer` information of the client certificate matches the information specified in `require issuer`.
- The `subject` information of the client certificate matches the information specified in `require cipher`.

You can log into TiDB only after all the above items are verified. Otherwise, the `ERROR 1045 (28000): Access denied` error is returned. You can use the following command to check the TLS version, the cipher algorithm and whether the current connection uses the certificate for the login.

Connect the MySQL client and execute the following statement:

```
\s
```

The output:

```
--------------
mysql Ver 15.1 Distrib 10.4.10-MariaDB, for Linux (x86_64) using readline 5.1

Connection id:    1
Current database: test
Current user:     root@127.0.0.1
SSL:              Cipher in use is TLS_AES_256_GCM_SHA384
```

Then execute the following statement:

```
show variables like '%ssl%';
```

The output:

```
+---------------+---------------------------------+
| Variable_name | Value                           |
+---------------+---------------------------------+
| ssl_cert      | /path/to/server-cert.pem        |
| ssl_ca        | /path/to/ca-cert.pem            |
| have_ssl      | YES                             |
| have_openssl  | YES                             |
| ssl_key       | /path/to/server-key.pem         |
+---------------+---------------------------------+
6 rows in set (0.067 sec)
```

4.3.5.4 Update and replace certificate

The key and certificate are updated regularly. The following sections introduce how to update the key and certificate.

The CA certificate is the basis for mutual verification between the client and server. To replace the CA certificate, generate a combined certificate that supports the authentication for both old and new certificates. On the client and server, first replace the CA certificate, then replace the client/server key and certificate.

4.3.5.4.1 Update CA key and certificate

1. Back up the old CA key and certificate (suppose that `ca-key.pem` is stolen):

```
mv ca-key.pem ca-key.old.pem && \
mv ca-cert.pem ca-cert.old.pem
```

2. Generate the new CA key:

```
sudo openssl genrsa 2048 > ca-key.pem
```

3. Generate the new CA certificate using the newly generated CA key:

```
sudo openssl req -new -x509 -nodes -days 365000 -key ca-key.pem -out ca
    ↪ -cert.new.pem
```

> **Note:**
>
> Generating the new CA certificate is to replace the keys and certificates on the client and server, and to ensure that online users are not affected. Therefore, the appended information in the above command must be consistent with the `require issuer` information.

4. Generate the combined CA certificate:

```
cat ca-cert.new.pem ca-cert.old.pem > ca-cert.pem
```

After the above operations, restart the TiDB server with the newly created combined CA certificate. Then the server accepts both the new and old CA certificates.

Also replace the old CA certificate with the combined certificate so that the client accepts both the old and new CA certificates.

4.3.5.4.2 Update client key and certificate

> **Note:**
>
> Perform the following steps **only after** you have replaced the old CA certificate on the client and server with the combined CA certificate.

1. Generate the new RSA key of the client:

```
sudo openssl req -newkey rsa:2048 -days 365000 -nodes -keyout client-
    ↪ key.new.pem -out client-req.new.pem && \
sudo openssl rsa -in client-key.new.pem -out client-key.new.pem
```

> **Note:**
>
> The above command is to replace the client key and certificate, and to ensure that the online users are not affected. Therefore, the appended information in the above command must be consistent with the `require subject` information.

2. Use the combined certificate and the new CA key to generate the new client certificate:

```
sudo openssl x509 -req -in client-req.new.pem -days 365000 -CA ca-cert.
    ↪ pem -CAkey ca-key.pem -set_serial 01 -out client-cert.new.pem
```

3. Make the client (for example, MySQL) connect TiDB with the new client key and certificate:

```
mysql -utest -h0.0.0.0 -P4000 --ssl-cert /path/to/client-cert.new.pem
    ↪ --ssl-key /path/to/client-key.new.pem --ssl-ca /path/to/ca-cert.
    ↪ pem
```

> **Note:**
>
> `/path/to/client-cert.new.pem`, `/path/to/client-key.new.pem`, and `/path/to/ca-cert.pem` specify the directory of the CA certificate, client key, and client certificate. You can replace them with your own directory.

4.3.5.4.3 Update the server key and certificate

1. Generate the new RSA key of the server:

```
sudo openssl req -newkey rsa:2048 -days 365000 -nodes -keyout server-
    ↪ key.new.pem -out server-req.new.pem && \
sudo openssl rsa -in server-key.new.pem -out server-key.new.pem
```

2. Use the combined CA certificate and the new CA key to generate the new server certificate:

```
sudo openssl x509 -req -in server-req.new.pem -days 365000 -CA ca-cert.
    ↪ pem -CAkey ca-key.pem -set_serial 01 -out server-cert.new.pem
```

3. Configure the TiDB server to use the new server key and certificate. See Configure TiDB server for details.

4.4 Transactions

4.4.1 Transactions

TiDB supports complete distributed transactions. Both optimistic transaction model and pessimistic transaction model(introduced in TiDB 3.0) are available. This document introduces transaction-related statements, explicit and implicit transactions, isolation levels, lazy check for constraints, and transaction sizes.

The common variables include `autocommit`, `tidb_disable_txn_auto_retry`, and `tidb_retry_limit`.

> **Note:**
>
> The `tidb_disable_txn_auto_retry` and `tidb_retry_limit` variables only apply to optimistic transactions, not to pessimistic transactions.

4.4.1.1 Common syntax

4.4.1.1.1 `BEGIN`, `START TRANSACTION` and `START TRANSACTION WITH CONSISTENT SNAPSHOT`

Syntax:

```
BEGIN;
```

```
START TRANSACTION;
```

```
START TRANSACTION WITH CONSISTENT SNAPSHOT;
```

All of the above three statements are used to start a transaction with the same effect. You can explicitly start a new transaction by executing one of these statements. If the current session is in the process of a transaction when one of these statements is executed, TiDB automatically commits the current transaction before starting a new transaction.

4.4.1.1.2 COMMIT

Syntax:

```
COMMIT;
```

You can use this statement to commit the current transaction, including all updates between `[BEGIN|START TRANSACTION]` and `COMMIT`.

4.4.1.1.3 ROLLBACK

Syntax:

```
ROLLBACK;
```

You can use this statement to roll back the current transaction and cancels all updates between `[BEGIN | START TRANSACTION]` and `ROLLBACK`.

4.4.1.2 Autocommit

Syntax:

```
SET autocommit = {0 | 1}
```

When `autocommit = 1` (default), the status of the current session is autocommit. That is, statements are automatically committed immediately following their execution.

When `autocommit = 0`, the status of the current session is non-autocommit. That is, statements are only committed when you manually execute the `COMMIT` statement.

> **Note:**
>
> Some statements are committed implicitly. For example, executing `[BEGIN| ↪ START TRANSACTION]` implicitly commits the last transaction and starts a new transaction. This behavior is required for MySQL compatibility. Refer to implicit commit for more details.

`autocommit` is also a system variable. You can update the current session or the Global value using the following variable assignment statement:

```
SET @@SESSION.autocommit = {0 | 1};
```

```
SET @@GLOBAL.autocommit = {0 | 1};
```

4.4.1.3 Explicit and implicit transaction

TiDB supports explicit transactions (use `[BEGIN|START TRANSACTION]` and `COMMIT` to define the start and end of the transaction) and implicit transactions (`SET autocommit = 1`).

If you set the value of `autocommit` to `1` and start a new transaction through the `[BEGIN|START TRANSACTION]` statement, the autocommit is disabled before `COMMIT` or `ROLLBACK` which makes the transaction becomes explicit.

For DDL statements, the transaction is committed automatically and does not support rollback. If you run the DDL statement while the current session is in the process of a transaction, the DDL statement is executed after the current transaction is committed.

4.4.1.4 Transaction isolation level

TiDB **only supports** `SNAPSHOT ISOLATION`. You can set the isolation level of the current session to `READ COMMITTED` using the following statement. However, TiDB is only compatible with the `READ COMMITTED` isolation level in syntax and transactions are still executed at the `SNAPSHOT ISOLATION` level.

```
SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

4.4.1.5 Lazy check of constraints

Lazy check means that by default TiDB will not check primary key or unique constraints when an `INSERT` statement is executed, but instead checks when the transaction is committed. In TiDB, the lazy check is performed for values written by ordinary `INSERT` statements.

For example:

```
CREATE TABLE t1 (id INT NOT NULL PRIMARY KEY);
INSERT INTO t1 VALUES (1);
START TRANSACTION;
INSERT INTO t1 VALUES (1); -- MySQL returns an error; TiDB returns success
    ↪ .
INSERT INTO t1 VALUES (2);
COMMIT; -- It is successfully committed in MySQL; TiDB returns an error
    ↪ and the transaction rolls back.
SELECT * FROM t1; -- MySQL returns 1 2; TiDB returns 1.
```

The lazy check is important because if you perform a unique constraint check on every `INSERT` statement in a transaction, it can cause high network overhead. A batch check when the transaction is committed can greatly improve performance.

Note:

This optimization does not take effect for `INSERT IGNORE` and `INSERT ON DUPLICATE KEY UPDATE`, only for normal `INSERT` statements. The behavior can also be disabled by setting `tidb_constraint_check_in_place=TRUE`.

4.4.1.6 Statement rollback

If you execute a statement within a transaction, the statement does not take effect when an error occurs.

```
begin;
insert into test values (1);
insert into tset values (2); // This statement does not take effect because "test" is misspelled as "tset".
insert into test values (3);
commit;
```

In the above example, the second `insert` statement fails, while the other two `insert` statements (1 & 3) can be successfully committed.

```
begin;
insert into test values (1);
insert into tset values (2); // This statement does not take effect because "test" is misspelled as "tset".
insert into test values (3);
rollback;
```

In the above example, the second `insert` statement fails, and this transaction does not insert any data into the database because `rollback` is called.

4.4.1.7 Transaction sizes

In TiDB, a transaction either too small or too large can impair the overall performance.

4.4.1.7.1 Small transactions

TiDB uses the default autocommit setting (that is, `autocommit = 1`), which automatically issues a commit when executing each SQL statement. Therefore, each of the following three statements is treated as a transaction:

```
UPDATE my_table SET a = 'new_value' WHERE id = 1;
UPDATE my_table SET a = 'newer_value' WHERE id = 2;
UPDATE my_table SET a = 'newest_value' WHERE id = 3;
```

In this case, the latency is increased because each statement, as a transaction, uses the two-phase commit which consumes more execution time.

To improve the execution efficiency, you can use an explicit transaction instead, that is, to execute the above three statements within a transaction:

```
START TRANSACTION;
UPDATE my_table SET a = 'new_value' WHERE id = 1;
UPDATE my_table SET a = 'newer_value' WHERE id = 2;
UPDATE my_table SET a = 'newest_value' WHERE id = 3;
COMMIT;
```

Similarly, it is recommended to execute `INSERT` statements within an explicit transaction.

> **Note:**
>
> The single-threaded workloads in TiDB might not fully use TiDB's distributed resources, so the performance of TiDB is lower than that of a single-instance deployment of MySQL. This difference is similar to the case of transactions with higher latency in TiDB.

4.4.1.7.2 Large transaction

Due to the requirement of the two-phase commit, a large transaction can lead to the following issues:

- OOM (Out of Memory) when excessive data is written in the memory
- More conflicts in the prewrite phase
- Long duration before transactions are actually committed

Therefore, TiDB intentionally imposes some limits on transaction sizes:

- The total number of SQL statements in a transaction is no more than 5,000 (default)
- Each key-value pair is no more than 6 MB
- The total number of key-value entries is no more than 300,000
- The total size of key-value entries is no more than 100 MB

For each transaction, it is recommended to keep the number of SQL statements between 100 to 500 to achieve an optimal performance.

4.4.2 TiDB Transaction Isolation Levels

Transaction isolation is one of the foundations of database transaction processing. Isolation is one of the four key properties of a transaction (commonly referred as ACID).

The SQL-92 standard defines four levels of transaction isolation: Read Uncommitted, Read Committed, Repeatable Read, and Serializable. See the following table for details:

Isolation Level	Dirty Write	Dirty Read	Fuzzy Read	Phantom
READ UNCOMMITTED	Not Possible	Possible	Possible	Possible
READ COMMITTED	Not Possible	Not possible	Possible	Possible
REPEATABLE READ	Not Possible	Not possible	Not possible	Possible
SERIALIZABLE	Not Possible	Not possible	Not possible	Not possible

TiDB implements Snapshot Isolation (SI) consistency, which it advertises as `REPEATABLE-READ` for compatibility with MySQL. This differs from the ANSI Repeatable Read isolation level and the MySQL Repeatable Read level.

> **Note:**
>
> In the default configuration of TiDB v3.0, the automatic transaction retry is disabled. For how this feature influences the isolation level and how to enable it, see automatic retry.

TiDB uses the Percolator transaction model. A global read timestamp is obtained when the transaction is started, and a global commit timestamp is obtained when the transaction is committed. The execution order of transactions is confirmed based on the timestamps. To know more about the implementation of TiDB transaction model, see MVCC in TiKV.

4.4.2.1 Repeatable Read

The Repeatable Read isolation level only sees data committed before the transaction begins, and it never sees either uncommitted data or changes committed during transaction execution by concurrent transactions. However, the transaction statement does see the effects of previous updates executed within its own transaction, even though they are not yet committed.

For transactions running on different nodes, the start and commit order depends on the order that the timestamp is obtained from PD.

Transactions of the Repeatable Read isolation level cannot concurrently update a same row. When committing, if the transaction finds that the row has been updated by another transaction after it starts, then the transaction rolls back and retries automatically. For example:

```
create table t1(id int);
insert into t1 values(0);

start transaction;          |             start transaction;
select * from t1;           |             select * from t1;
update t1 set id=id+1;      |             update t1 set id=id+1;
commit;                     |
                            |             commit; -- the transaction commit
                               ↪ fails and rolls back
```

4.4.2.1.1 Difference between TiDB and ANSI Repeatable Read

The Repeatable Read isolation level in TiDB differs from ANSI Repeatable Read isolation level, though they sharing the same name. According to the standard described in the A Critique of ANSI SQL Isolation Levels paper, TiDB implements the Snapshot Isolation level. This isolation level does not allow strict phantoms (A3) but allows broad phantoms (P3) and write skews. In contrast, the ANSI Repeatable Read isolation level allows phantom reads but does not allow write skews.

4.4.2.1.2 Difference between TiDB and MySQL Repeatable Read

The Repeatable Read isolation level in TiDB differs from that in MySQL. The MySQL Repeatable Read isolation level does not check whether the current version is visible when updating, which means it can continue to update even if the row has been updated after the transaction starts. In contrast, if the row has been updated after the transaction starts, the TiDB transaction is rolled back and retried. Transaction Retries in TiDB might fail, leading to a final failure of the transaction, while in MySQL the updating transaction can be successful.

The MySQL Repeatable Read isolation level is not the snapshot isolation level. The consistency of MySQL Repeatable Read isolation level is weaker than both the snapshot isolation level and TiDB Repeatable Read isolation level.

4.4.3 TiDB Optimistic Transaction Model

This document introduces the principles of TiDB's optimistic transaction model. This document assumes that you have a basic understanding of TiDB architecture, Percolator, and the ACID properties of transactions.

In TiDB's optimistic transaction model, write-write conflicts are detected only at the two-phase transactional commit.

Note:

Starting from v3.0.8, TiDB uses the pessimistic transaction model by default. However, this does not affect your clusters if you upgrading from v3.0.7 or earlier to v3.0.8 (and later). In other words, **only newly created clusters default to using the pessimistic transaction model**.

4.4.3.1 Principles of optimistic transactions

TiDB adopts Google's Percolator transaction model, a variant of two-phase commit (2PC) to ensure the correct completion of a distributed transaction. The procedure is as follows:

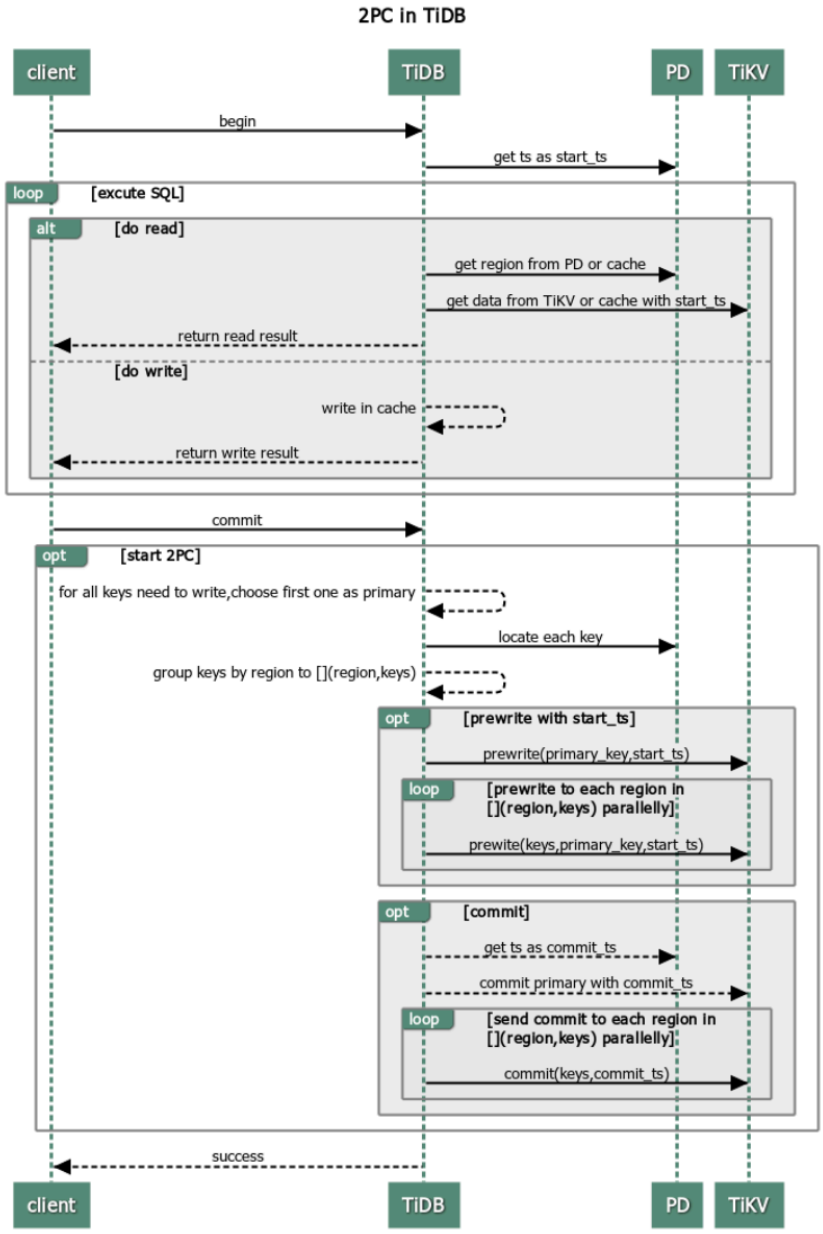

Figure 266: 2PC in TiDB

1. The client begins a transaction.

   TiDB receives the start version number (monotonically increasing in time and globally unique) from PD and mark it as `start_ts`.

2. The client issues a read request.

   1. TiDB receives routing information (how data is distributed among TiKV nodes) from PD.

2. TiDB receives the data of the `start_ts` version from TiKV.

3. The client issues a write request.

   TiDB checks whether the written data satisfies consistency constraints (to ensure the data types are correct and the unique index is met etc.) **Valid data is stored in the memory**.

4. The client issues a commit request.

5. TiDB begins 2PC to ensure the atomicity of distributed transactions and persist data in store.

   1. TiDB selects a Primary Key from the data to be written.
   2. TiDB receives the information of Region distribution from PD, and groups all keys by Region accordingly.
   3. TiDB sends prewrite requests to all TiKV nodes involved. Then, TiKV checks whether there are conflict or expired versions. Valid data is locked.
   4. TiDB receives all responses in the prewrite phase and the prewrite is successful.
   5. TiDB receives a commit version number from PD and marks it as `commit_ts`.
   6. TiDB initiates the second commit to the TiKV node where Primary Key is located. TiKV checks the data, and clean the locks left in the prewrite phase.
   7. TiDB receives the message that reports the second phase is successfully finished.

6. TiDB returns a message to inform the client that the transaction is successfully committed.

7. TiDB asynchronously cleans the locks left in this transaction.

4.4.3.2 Advantages and disadvantages

From the process of transactions in TiDB above, it is clear that TiDB transactions have the following advantages:

- Simple to understand
- Implement cross-node transaction based on single-row transaction
- Decentralized lock management

However, TiDB transactions also have the following disadvantages:

- Transaction latency due to 2PC
- In need of a centralized version manager
- OOM (out of memory) when extensive data is written in the memory

To avoid potential problems in application, refer to transaction sizes to see more details.

4.4.3.3 Transaction retries

TiDB uses optimistic concurrency control by default whereas MySQL applies pessimistic concurrency control. This means that MySQL checks for conflicts during the execution of SQL statements, so there are few errors reported in heavy contention scenarios. For the convenience of MySQL users, TiDB provides a retry function that runs inside a transaction.

4.4.3.3.1 Automatic retry

If there is a conflict, TiDB retries the write operations automatically. You can set `tidb_disable_txn_auto_retry` and `tidb_retry_limit` to enable or disable this default function:

```
### Whether to disable automatic retry. ("on" by default)
tidb_disable_txn_auto_retry = off
### Set the maximum number of the retires. ("10" by default)
### When "tidb_retry_limit = 0", automatic retry is completely disabled.
tidb_retry_limit = 10
```

You can enable the automatic retry in either session level or global level:

1. Session level:

```
set @@tidb_disable_txn_auto_retry = off;
```

```
set @@tidb_retry_limit = 10;
```

2. Global level:

```
set @@global.tidb_disable_txn_auto_retry = off;
```

```
set @@global.tidb_retry_limit = 10;
```

> **Note:**
>
> The `tidb_retry_limit` variable decides the maximum number of retries. When this variable is set to `0`, none of the transactions automatically retries, including the implicit single statement transactions that are automatically committed. This is the way to completely disable the automatic retry mechanism in TiDB. After the automatic retry is disabled, all conflicting transactions report failures (includes the `try again later` string) to the application layer in the fastest way.

4.4.3.3.2 Limits of retry

By default, TiDB will not retry transactions because this might lead to lost updates and damaged `REPEATABLE READ` isolation.

The reason can be observed from the procedures of retry:

1. Allocate a new timestamp and mark it as `start_ts`.
2. Retry the SQL statements that contain write operations.
3. Implement the two-phase commit.

In Step 2, TiDB only retries SQL statements that contain write operations. However, during retrying, TiDB receives a new version number to mark the beginning of the transaction. This means that TiDB retries SQL statements with the data in the new `start_ts` version. In this case, if the transaction updates data using other query results, the results might be inconsistent because the `REPEATABLE READ` isolation is violated.

If your application can tolerate lost updates, and does not require `REPEATABLE READ` isolation consistency, you can enable this feature by setting `tidb_disable_txn_auto_retry = off`.

4.4.3.4 Conflict detection

For the optimistic transaction, it is important to detect whether there are write-write conflicts in the underlying data. Although TiKV reads data for detection **in the prewrite phase**, a conflict pre-detection is also performed in the TiDB clusters to improve the efficiency.

Because TiDB is a distributed database, the conflict detection in the memory is performed in two layers:

- The TiDB layer. If a write-write conflict in the instance is observed after the primary write is issued, it is unnecessary to issue the subsequent writes to the TiKV layer.
- The TiKV layer. TiDB instances are unaware of each other, which means they cannot confirm whether there are conflicts or not. Therefore, the conflict detection is mainly performed in the TiKV layer.

The conflict detection in the TiDB layer is disabled by default. The specific configuration items are as follows:

```
[txn-local-latches]
### Whether to enable the latches for transactions. Recommended
### to use latches when there are many local transaction conflicts.
### ("false" by default)
enabled = false
### Controls the number of slots corresponding to Hash. ("204800" by default
    ↪ )
```

```
### It automatically adjusts upward to an exponential multiple of 2.
### Each slot occupies 32 Bytes of memory. If set too small,
### it might result in slower running speed and poor performance
### when data writing covers a relatively large range.
capacity = 2048000
```

The value of the `capacity` configuration item mainly affects the accuracy of conflict detection. During conflict detection, only the hash value of each key is stored in the memory. Because the probability of collision when hashing is closely related to the probability of misdetection, you can configure `capacity` to controls the number of slots and enhance the accuracy of conflict detection.

- The smaller the value of `capacity`, the smaller the occupied memory and the greater the probability of misdetection.
- The larger the value of `capacity`, the larger the occupied memory and the smaller the probability of misdetection.

When you confirm that there is no write-write conflict in the upcoming transactions (such as importing data), it is recommended to disable the function of conflict detection.

TiKV also uses a similar mechanism to detect conflicts, but the conflict detection in the TiKV layer cannot be disabled. You can only configure `scheduler-concurrency` to control the number of slots that defined by the modulo operation:

```
### Controls the number of slots. ("2048000" by default)
scheduler-concurrency = 2048000
```

In addition, TiKV supports monitoring the time spent on waiting latches in scheduler.

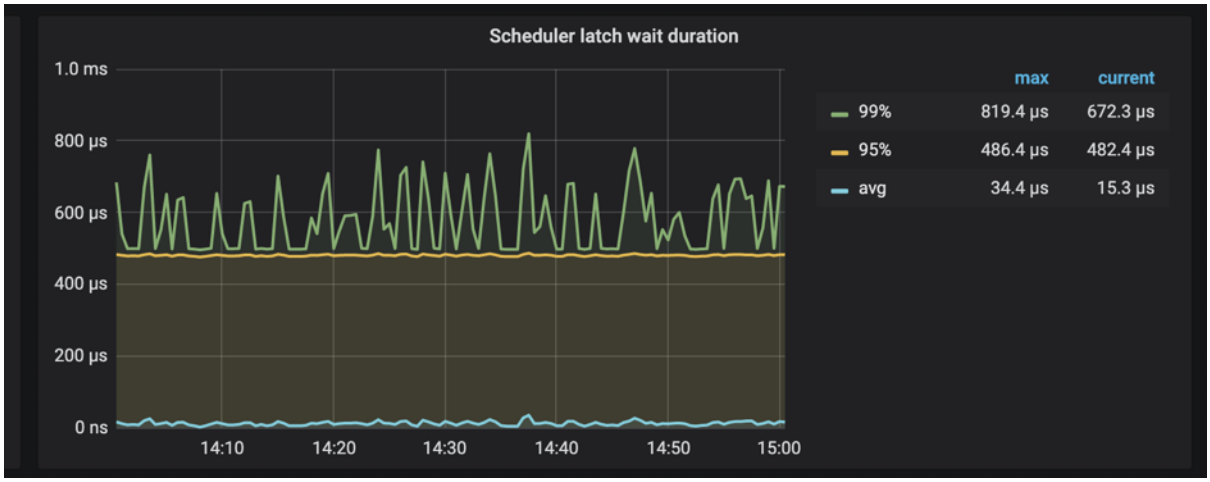

Figure 267: Scheduler latch wait duration

When `Scheduler latch wait duration` is high and there is no slow writes, it can be safely concluded that there are many write conflicts at this time.

4.4.4 TiDB Pessimistic Transaction Model

In versions before 3.0.8, TiDB implements the optimistic transaction mode by default, in which the transaction commit might fail because of transaction conflict. To make sure that the commit succeeds, you need to modify the application and add an automatic retry mechanism. You can avoid this issue by using the pessimistic transaction mode of TiDB.

4.4.4.1 Usage

To apply the pessimistic transaction mode, choose any of the following three methods that suits your needs:

- Execute the `BEGIN PESSIMISTIC;` statement to allow the transaction to apply the pessimistic transaction mode. You can write it in comment style as `BEGIN /*!90000 PESSIMISTIC */;` to make it compatible with the MySQL syntax.
- Execute the `set @@tidb_txn_mode = 'pessimistic';` statement to allow all the explicit transactions (namely non-autocommit transactions) processed in this session to apply the pessimistic transaction mode.
- Execute the `set @@global.tidb_txn_mode = 'pessimistic';` statement to allow all newly created sessions of the entire cluster to apply the pessimistic transaction mode to execute explicit transactions.

After you set `global.tidb_txn_mode` to `pessimistic`, the pessimistic transaction mode is applied by default. To apply the optimistic transaction mode to the transaction, you can use any of the following three methods:

- Execute the `BEGIN OPTIMISTIC;` statement to allow the transaction to apply the optimistic transaction mode. You can write it in comment style as `BEGIN /*!90000 OPTIMISTIC */;` to make it compatible with the MySQL syntax.
- Execute the `set @@tidb_txn_mode = 'optimistic';` statement to allow all the transactions processed in this session to apply the optimistic transaction mode.
- Execute the `set @@global.tidb_txn_mode = 'optimistic;'` or `set @@global.tidb_txn_mode = '';` to allow all newly created sessions of the entire cluster to apply the optimistic transaction mode to the transactions.

The `BEGIN PESSIMISTIC;` and `BEGIN OPTIMISTIC;` statements take precedence over the `tidb_txn_mode` system variable. Transactions started with these two statements ignore the system variable and support using both the pessimistic and optimistic transaction modes.

To disable the pessimistic transaction mode, modify the configuration file and add `enable = false` to the `[pessimistic-txn]` category.

4.4.4.2 Behaviors

Pessimistic transactions in TiDB behave similarly to those in MySQL. See the minor differences in Difference with MySQL InnoDB.

- When you perform the `SELECT FOR UPDATE` statement, transactions read the last committed data and apply a pessimistic lock on the data being read.
- When you perform the `UPDATE`, `DELETE` or `INSERT` statement, transactions read the last committed data to execute on them and apply a pessimistic lock on the modified data.
- When a pessimistic lock is applied on a row of data, other write transactions attempting to modify the data are blocked and have to wait for the lock to be released.
- When a pessimistic lock is applied on a row of data, other transactions attempting to read the data are not blocked and can read the committed data.
- All the locks are released when the transaction is committed or rolled back.
- When multiple transactions wait for the same lock to be released, the lock is acquired in the order of the `start ts` of the transactions as much as possible; however, the order cannot be strictly guaranteed.
- Deadlocks in concurrent transactions can be detected by the deadlock detector. The detector randomly terminates one of the transactions, and a MySQL-compatible error code `1213` is returned.
- TiDB supports both the optimistic transaction mode and pessimistic transaction mode in the same cluster. You can specify either mode for transaction execution.
- TiDB sets the lock wait timeout time by the `innodb_lock_wait_timeout` variable. After the lock times out, a MySQL-compatible error code `1205` is returned.

4.4.4.3 Difference with MySQL InnoDB

1. When TiDB executes DML or `SELECT FOR UPDATE` statements that use range in the WHERE clause, the concurrent `INSERT` statements within the range are not blocked.

   By implementing Gap Lock, InnoDB blocks the execution of concurrent `INSERT` statements within the range. It is mainly used to support statement-based binlog. Therefore, some applications lower the isolation level to READ COMMITTED to avoid concurrency performance problems caused by Gap Lock. TiDB does not support Gap Lock, so there is no need to pay the concurrency performance cost.

2. TiDB does not support `SELECT LOCK IN SHARE MODE`.

   When `SELECT LOCK IN SHARE MODE` is executed, it has the same effect as that without the lock, so the read or write operation of other transactions is not blocked.

3. DDL may result in failure of the pessimistic transaction commit.

   When DDL is executed in MySQL, it might be blocked by the transaction that is being executed. However, in this scenario, the DDL operation is not blocked in TiDB, which leads to failure of the pessimistic transaction commit: `ERROR 1105 (HY000): Information schema is changed. [try again later]`.

4. After executing `START TRANSACTION WITH CONSISTENT SNAPSHOT`, MySQL can still read the tables that are created later in other transactions, while TiDB cannot.

5. The autocommit transactions do not support the pessimistic locking.

   None of the autocommit statements acquire the pessimistic lock. These statements do not display any difference in the user side, because the nature of pessimistic transactions is to turn the retry of the whole transaction into a single DML retry. The autocommit transactions automatically retry even when TiDB closes the retry, which has the same effect as pessimistic transactions.

   The autocommit `SELECT FOR UPDATE` statement does not wait for lock, either.

4.4.4.4 FAQ

1. The TiDB log shows `pessimistic write conflict, retry statement`.

   When a write conflict occurs, the optimistic transaction is terminated directly, but the pessimistic transaction retries the statement with the latest data until there is no write conflict. The log prints this entry with each retry, so there is no need for extra attention.

2. When DML is executed, an error `pessimistic lock retry limit reached` is returned.

   In the pessimistic transaction mode, every statement has a retry limit. This error is returned when the retry times of write conflict exceeds the limit. The default retry limit is `256`. To change the limit, modify the `max-retry-limit` under the `[pessimistic-txn]` category in the TiDB configuration file.

3. The execution time limit for pessimistic transactions.

   The execution time of transactions cannot exceed the limit of `tikv_gc_life_time`. In addition, the pessimistic transactions have a TTL (Time to Live) limit of 10 minutes, so the pessimistic transactions that execute over 10 minutes might fail to commit.

4.5 System Databases

4.5.1 TiDB System Tables

This document introduces TiDB system tables.

4.5.1.1 Grant system tables

These system tables contain grant information about user accounts and their privileges:

- `user`: user accounts, global privileges, and other non-privilege columns
- `db`: database-level privileges
- `tables_priv`: table-level privileges
- `columns_priv`: column-level privileges

4.5.1.2 Server-side help system tables

Currently, the `help_topic` is NULL.

4.5.1.3 Statistics system tables

- `stats_buckets`: the buckets of statistics
- `stats_histograms`: the histograms of statistics
- `stats_meta`: the meta information of tables, such as the total number of rows and updated rows

4.5.1.4 GC worker system tables

- `gc_delete_range`: to record the data to be deleted

4.5.1.5 Miscellaneous system tables

- `GLOBAL_VARIABLES`: global system variable table
- `tidb`: to record the version information when TiDB executes `bootstrap`

4.5.2 Information Schema

As part of MySQL compatibility, TiDB supports a number of `INFORMATION_SCHEMA` ↪ tables. Many of these tables also have a corresponding `SHOW` command. The benefit of querying `INFORMATION_SCHEMA` is that it is possible to join between tables.

4.5.2.1 Fully Supported Information Schema Tables

4.5.2.1.1 ANALYZE_STATUS table

The ANALYZE_STATUS table provides information about the running tasks that collect statistics and a limited number of history tasks.

```
select * from `ANALYZE_STATUS`;
```

```
+--
    ↪ -----------+-----------+---------------+------------------+----------------+
    ↪
| TABLE_SCHEMA | TABLE_NAME | PARTITION_NAME | JOB_INFO  | PROCESSED_ROWS |
    ↪ START_TIME      | STATE   |
+--
    ↪ -----------+-----------+---------------+------------------+----------------+
    ↪
| test         | t          |               | analyze index idx | 2              |
    ↪ 2019-06-21 19:51:14 | finished |
| test         | t          |               | analyze columns | 2              |
    ↪ 2019-06-21 19:51:14 | finished |
| test         | t1         | p0            | analyze columns | 0              |
    ↪ 2019-06-21 19:51:15 | finished |
| test         | t1         | p3            | analyze columns | 0              |
    ↪ 2019-06-21 19:51:15 | finished |
| test         | t1         | p1            | analyze columns | 0              |
    ↪ 2019-06-21 19:51:15 | finished |
| test         | t1         | p2            | analyze columns | 1              |
    ↪ 2019-06-21 19:51:15 | finished |
+--
    ↪ -----------+-----------+---------------+------------------+----------------+
    ↪
6 rows in set
```

4.5.2.1.2 CHARACTER_SETS table

The CHARACTER_SETS table provides information about character sets. Currently, TiDB only supports some of the character sets.

```
SELECT * FROM character_sets;
```

```
+--------------------+----------------------+--------------+--------+
| CHARACTER_SET_NAME | DEFAULT_COLLATE_NAME | DESCRIPTION | MAXLEN |
+--------------------+----------------------+--------------+--------+
| utf8               | utf8_bin             | UTF-8 Unicode |      3 |
| utf8mb4            | utf8mb4_bin          | UTF-8 Unicode |      4 |
| ascii              | ascii_bin            | US ASCII      |      1 |
| latin1             | latin1_bin           | Latin1        |      1 |
```

```
| binary           | binary              | binary      |     1 |
+------------------+---------------------+-------------+-------+
5 rows in set (0.00 sec)
```

4.5.2.1.3 COLLATIONS table

The `COLLATIONS` table provides a list of collations that correspond to character sets in the `CHARACTER_SETS` table. Currently this table is included only for compatibility with MySQL, as TiDB only supports binary collation:

```
SELECT * FROM collations WHERE character_set_name='utf8mb4';
```

```
+--
   ↪ --------------------+--------------------+------+-----------+-------------+--
   ↪
| COLLATION_NAME      | CHARACTER_SET_NAME | ID | IS_DEFAULT | IS_COMPILED |
   ↪  SORTLEN |
+--
   ↪ --------------------+--------------------+------+-----------+-------------+--
   ↪
| utf8mb4_general_ci  | utf8mb4            |   45 | Yes        | Yes         |
   ↪      1 |
| utf8mb4_bin         | utf8mb4            |   46 |            | Yes         |
   ↪      1 |
| utf8mb4_unicode_ci  | utf8mb4            |  224 |            | Yes         |
   ↪      1 |
| utf8mb4_icelandic_ci | utf8mb4           |  225 |            | Yes         |
   ↪      1 |
| utf8mb4_latvian_ci  | utf8mb4            |  226 |            | Yes         |
   ↪      1 |
| utf8mb4_romanian_ci | utf8mb4            |  227 |            | Yes         |
   ↪      1 |
| utf8mb4_slovenian_ci | utf8mb4           |  228 |            | Yes         |
   ↪      1 |
| utf8mb4_polish_ci   | utf8mb4            |  229 |            | Yes         |
   ↪      1 |
| utf8mb4_estonian_ci | utf8mb4            |  230 |            | Yes         |
   ↪      1 |
| utf8mb4_spanish_ci  | utf8mb4            |  231 |            | Yes         |
   ↪      1 |
| utf8mb4_swedish_ci  | utf8mb4            |  232 |            | Yes         |
   ↪      1 |
| utf8mb4_turkish_ci  | utf8mb4            |  233 |            | Yes         |
   ↪      1 |
```

```
| utf8mb4_czech_ci    | utf8mb4          |  234 |          | Yes       |
   ↪       1 |
| utf8mb4_danish_ci   | utf8mb4          |  235 |          | Yes       |
   ↪       1 |
| utf8mb4_lithuanian_ci | utf8mb4        |  236 |          | Yes       |
   ↪       1 |
| utf8mb4_slovak_ci   | utf8mb4          |  237 |          | Yes       |
   ↪       1 |
| utf8mb4_spanish2_ci | utf8mb4          |  238 |          | Yes       |
   ↪       1 |
| utf8mb4_roman_ci    | utf8mb4          |  239 |          | Yes       |
   ↪       1 |
| utf8mb4_persian_ci  | utf8mb4          |  240 |          | Yes       |
   ↪       1 |
| utf8mb4_esperanto_ci | utf8mb4         |  241 |          | Yes       |
   ↪       1 |
| utf8mb4_hungarian_ci | utf8mb4         |  242 |          | Yes       |
   ↪       1 |
| utf8mb4_sinhala_ci  | utf8mb4          |  243 |          | Yes       |
   ↪       1 |
| utf8mb4_german2_ci  | utf8mb4          |  244 |          | Yes       |
   ↪       1 |
| utf8mb4_croatian_ci | utf8mb4          |  245 |          | Yes       |
   ↪       1 |
| utf8mb4_unicode_520_ci | utf8mb4       |  246 |          | Yes       |
   ↪       1 |
| utf8mb4_vietnamese_ci | utf8mb4        |  247 |          | Yes       |
   ↪       1 |
+--
   ↪ ---------------------+--------------------+------+-----------+-------------+--
   ↪
26 rows in set (0.00 sec)
```

4.5.2.1.4 COLLATION_CHARACTER_SET_APPLICABILITY table

The `COLLATION_CHARACTER_SET_APPLICABILITY` table maps collations to the applicable character set name. Similar to the `COLLATIONS` table, it is included only for compatibility with MySQL:

```
SELECT * FROM collation_character_set_applicability WHERE
   ↪ character_set_name='utf8mb4';
```

```
+-----------------------+--------------------+
| COLLATION_NAME        | CHARACTER_SET_NAME |
+-----------------------+--------------------+
```

```
| utf8mb4_general_ci  | utf8mb4          |
| utf8mb4_bin         | utf8mb4          |
| utf8mb4_unicode_ci  | utf8mb4          |
| utf8mb4_icelandic_ci | utf8mb4         |
| utf8mb4_latvian_ci  | utf8mb4          |
| utf8mb4_romanian_ci | utf8mb4          |
| utf8mb4_slovenian_ci | utf8mb4         |
| utf8mb4_polish_ci   | utf8mb4          |
| utf8mb4_estonian_ci | utf8mb4          |
| utf8mb4_spanish_ci  | utf8mb4          |
| utf8mb4_swedish_ci  | utf8mb4          |
| utf8mb4_turkish_ci  | utf8mb4          |
| utf8mb4_czech_ci    | utf8mb4          |
| utf8mb4_danish_ci   | utf8mb4          |
| utf8mb4_lithuanian_ci | utf8mb4        |
| utf8mb4_slovak_ci   | utf8mb4          |
| utf8mb4_spanish2_ci | utf8mb4          |
| utf8mb4_roman_ci    | utf8mb4          |
| utf8mb4_persian_ci  | utf8mb4          |
| utf8mb4_esperanto_ci | utf8mb4         |
| utf8mb4_hungarian_ci | utf8mb4         |
| utf8mb4_sinhala_ci  | utf8mb4          |
| utf8mb4_german2_ci  | utf8mb4          |
| utf8mb4_croatian_ci | utf8mb4          |
| utf8mb4_unicode_520_ci | utf8mb4       |
| utf8mb4_vietnamese_ci | utf8mb4        |
+------------------------+--------------------+
26 rows in set (0.00 sec)
```

4.5.2.1.5 COLUMNS table

The `COLUMNS` table provides detailed information about columns in tables:

```
CREATE TABLE test.t1 (a int);
```

```
1 row in set (0.01 sec)
```

```
SELECT * FROM information_schema.columns WHERE table_schema='test' AND
   ↪ TABLE_NAME='t1';
```

```
*************************** 1. row ***************************
          TABLE_CATALOG: def
           TABLE_SCHEMA: test
             TABLE_NAME: t1
            COLUMN_NAME: a
```

```
       ORDINAL_POSITION: 1
         COLUMN_DEFAULT: NULL
            IS_NULLABLE: YES
              DATA_TYPE: int
CHARACTER_MAXIMUM_LENGTH: NULL
  CHARACTER_OCTET_LENGTH: NULL
       NUMERIC_PRECISION: 11
           NUMERIC_SCALE: 0
      DATETIME_PRECISION: NULL
      CHARACTER_SET_NAME: NULL
          COLLATION_NAME: NULL
             COLUMN_TYPE: int(11)
              COLUMN_KEY:
                   EXTRA:
              PRIVILEGES: select,insert,update,references
          COLUMN_COMMENT:
   GENERATION_EXPRESSION:
1 row in set (0.01 sec)
```

The description of columns in the `COLUMNS` table is as follows:

- `TABLE_CATALOG`: The name of the catalog to which the table with the column belongs. The value is always `def`.
- `TABLE_SCHEMA`: The name of the schema in which the table with the column is located.
- `TABLE_NAME`: The name of the table with the column.
- `COLUMN_NAME`: The name of the column.
- `ORDINAL_POSITION`: The position of the column in the table.
- `COLUMN_DEFAULT`: The default value of the column. If the explicit default value is `NULL`, or if the column definition does not include the `default` clause, this value is `NULL`.
- `IS_NULLABLE`: Whether the column is nullable. If the column can store null values, this value is `YES`; otherwise, it is `NO`.
- `DATA_TYPE`: The type of data in the column.
- `CHARACTER_MAXIMUM_LENGTH`: For string columns, the maximum length in characters.
- `CHARACTER_OCTET_LENGTH`: For string columns, the maximum length in bytes.
- `NUMERIC_PRECISION`: The numeric precision of a number-type column.
- `NUMERIC_SCALE`: The numeric scale of a number-type column.
- `DATETIME_PRECISION`: For time-type columns, the fractional seconds precision.
- `CHARACTER_SET_NAME`: The name of the character set of a string column.
- `COLLATION_NAME`: The name of the collation of a string column.
- `COLUMN_TYPE`: The column type.
- `COLUMN_KEY`: Whether this column is indexed. This field might have the following values:
  - Empty: This column is not indexed, or this column is indexed and is the second column in a multi-column non-unique index.

  - PRI: This column is the primary key or one of multiple primary keys.
  - UNI: This column is the first column of the unique index.
  - MUL: The column is the first column of a non-unique index, in which a given value is allowed to occur for multiple times.

- EXTRA: Any additional information of the given column.
- PRIVILEGES: The privilege that the current user has on this column. Currently, this value is fixed in TiDB, and is always `select,insert,update,references`.
- COLUMN_COMMENT: Comments contained in the column definition.
- GENERATION_EXPRESSION: For generated columns, this value displays the expression used to calculate the column value. For non-generated columns, the value is empty.

The corresponding SHOW statement is as follows:

```
SHOW COLUMNS FROM t1 FROM test;
```

```
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| a     | int(11) | YES  |     | NULL    |       |
+-------+---------+------+-----+---------+-------+
1 row in set (0.00 sec)
```

4.5.2.1.6 ENGINES table

The ENGINES table provides information about storage engines. For compatibility, TiDB will always describe InnoDB as the only supported engine. In addition, other column values in the ENGINES table are also fixed values.

```
SELECT * FROM engines;
```

```
*************************** 1. row ***************************
      ENGINE: InnoDB
     SUPPORT: DEFAULT
     COMMENT: Supports transactions, row-level locking, and foreign keys
TRANSACTIONS: YES
          XA: YES
  SAVEPOINTS: YES
1 row in set (0.00 sec)
```

The description of columns in the ENGINES table is as follows:

- ENGINES: The name of the storage engine.
- SUPPORT: The level of support that the server has on the storage engine. In TiDB, the value is always DEFAULT.

- `COMMENT`: The brief comment on the storage engine.
- `TRANSACTIONS`: Whether the storage engine supports transactions.
- `XA`: Whether the storage engine supports XA transactions.
- `SAVEPOINTS`: Whether the storage engine supports `savepoints`.

4.5.2.1.7 KEY_COLUMN_USAGE table

The `KEY_COLUMN_USAGE` table describes the key constraints of the columns, such as the primary key constraint:

```
SELECT * FROM key_column_usage WHERE table_schema='mysql' and table_name='
    ↪ user';
```

```
*************************** 1. row ***************************
          CONSTRAINT_CATALOG: def
           CONSTRAINT_SCHEMA: mysql
             CONSTRAINT_NAME: PRIMARY
               TABLE_CATALOG: def
                TABLE_SCHEMA: mysql
                  TABLE_NAME: user
                 COLUMN_NAME: Host
            ORDINAL_POSITION: 1
POSITION_IN_UNIQUE_CONSTRAINT: NULL
     REFERENCED_TABLE_SCHEMA: NULL
       REFERENCED_TABLE_NAME: NULL
      REFERENCED_COLUMN_NAME: NULL
*************************** 2. row ***************************
          CONSTRAINT_CATALOG: def
           CONSTRAINT_SCHEMA: mysql
             CONSTRAINT_NAME: PRIMARY
               TABLE_CATALOG: def
                TABLE_SCHEMA: mysql
                  TABLE_NAME: user
                 COLUMN_NAME: User
            ORDINAL_POSITION: 2
POSITION_IN_UNIQUE_CONSTRAINT: NULL
     REFERENCED_TABLE_SCHEMA: NULL
       REFERENCED_TABLE_NAME: NULL
      REFERENCED_COLUMN_NAME: NULL
2 rows in set (0.00 sec)
```

The description of columns in the `KEY_COLUMN_USAGE` table is as follows:

- `CONSTRAINT_CATALOG`: The name of the catalog to which the constraint belongs. The value is always `def`.

- `CONSTRAINT_SCHEMA`: The name of the schema to which the constraint belongs.
- `CONSTRAINT_NAME`: The name of the constraint.
- `TABLE_CATALOG`: The name of the catalog to which the table belongs. The value is always `def`.
- `TABLE_SCHEMA`: The name of the schema to which the table belongs.
- `TABLE_NAME`: The name of the table with constraints.
- `COLUMN_NAME`: The name of the column with constraints.
- `ORDINAL_POSITION`: The position of the column in the constraint, rather than in the table. The position number starts from `1`.
- `POSITION_IN_UNIQUE_CONSTRAINT`: The unique constraint and the primary key constraint are empty. For foreign key constraints, this column is the position of the referenced table's key.
- `REFERENCED_TABLE_SCHEMA`: The name of the schema referenced by the constraint. Currently in TiDB, the value of this column in all constraints is `nil`, except for the foreign key constraint.
- `REFERENCED_TABLE_NAME`: The name of the table referenced by the constraint. Currently in TiDB, the value of this column in all constraints is `nil`, except for the foreign key constraint.
- `REFERENCED_COLUMN_NAME`: The name of the column referenced by the constraint. Currently in TiDB, the value of this column in all constraints is `nil`, except for the foreign key constraint.

4.5.2.1.8 PROCESSLIST table

`PROCESSLIST`, just like `show processlist`, is used to view the requests that are being handled.

The `PROCESSLIST` table has a `MEM` column New in v3.0.5 that `show processlist` does not have. `MEM` means the occupied memory of the requests being handled, and its unit is `byte`.

```
+--
   ↪ --+------+------+--------------------+---------+------+-------+----------------
   ↪
| ID | USER | HOST | DB             | COMMAND | TIME | STATE | INFO
   ↪                | MEM |
+--
   ↪ --+------+------+--------------------+---------+------+-------+----------------
   ↪
| 1  | root | ::1 | INFORMATION_SCHEMA | Query | 0 | 2    | select * from
   ↪ PROCESSLIST | 0 |
+--
   ↪ --+------+------+--------------------+---------+------+-------+----------------
   ↪
```

4.5.2.1.9 SCHEMATA table

The SCHEMATA table provides information about databases. The table data is equivalent to the result of the SHOW DATABASES statement:

```
select * from SCHEMATA;
```

```
*************************** 1. row ***************************
              CATALOG_NAME: def
               SCHEMA_NAME: INFORMATION_SCHEMA
DEFAULT_CHARACTER_SET_NAME: utf8mb4
    DEFAULT_COLLATION_NAME: utf8mb4_bin
                  SQL_PATH: NULL
*************************** 2. row ***************************
              CATALOG_NAME: def
               SCHEMA_NAME: mysql
DEFAULT_CHARACTER_SET_NAME: utf8mb4
    DEFAULT_COLLATION_NAME: utf8mb4_bin
                  SQL_PATH: NULL
*************************** 3. row ***************************
              CATALOG_NAME: def
               SCHEMA_NAME: PERFORMANCE_SCHEMA
DEFAULT_CHARACTER_SET_NAME: utf8mb4
    DEFAULT_COLLATION_NAME: utf8mb4_bin
                  SQL_PATH: NULL
*************************** 4. row ***************************
              CATALOG_NAME: def
               SCHEMA_NAME: test
DEFAULT_CHARACTER_SET_NAME: utf8mb4
    DEFAULT_COLLATION_NAME: utf8mb4_bin
                  SQL_PATH: NULL
4 rows in set (0.00 sec)
```

4.5.2.1.10 SESSION_VARIABLES table

The SESSION_VARIABLES table provides information about session variables. The table data is similar to the result of the SHOW SESSION VARIABLES statement:

```
SELECT * FROM session_variables LIMIT 10;
```

```
+-------------------------------+---------------------+
| VARIABLE_NAME                 | VARIABLE_VALUE      |
+-------------------------------+---------------------+
| max_write_lock_count          | 18446744073709551615 |
| server_id_bits                | 32                  |
| net_read_timeout              | 30                  |
```

```
| innodb_online_alter_log_max_size | 134217728           |
| innodb_optimize_fulltext_only | OFF                 |
| max_join_size                 | 18446744073709551615 |
| innodb_read_io_threads        | 4                   |
| session_track_gtids           | OFF                 |
| have_ssl                      | DISABLED            |
| max_binlog_cache_size         | 18446744073709547520 |
+-------------------------------+----------------------+
10 rows in set (0.00 sec)
```

4.5.2.2 SLOW_QUERY table

The SLOW_QUERY table provides the slow query information, which is the parsing result of the TiDB slow log file. The column names in the table are corresponding to the field names in the slow log. For how to use this table to identify problematic statements and improve query performance of the SQL engine, see Slow Query Log Document.

```
mysql> desc information_schema.slow_query;
+---------------+---------------------+------+------+---------+-------+
| Field         | Type                | Null | Key  | Default | Extra |
+---------------+---------------------+------+------+---------+-------+
| Time          | timestamp unsigned  | YES  |      | NULL    |       |
| Txn_start_ts  | bigint(20) unsigned | YES  |      | NULL    |       |
| User          | varchar(64)         | YES  |      | NULL    |       |
| Host          | varchar(64)         | YES  |      | NULL    |       |
| Conn_ID       | bigint(20) unsigned | YES  |      | NULL    |       |
| Query_time    | double unsigned     | YES  |      | NULL    |       |
| Process_time  | double unsigned     | YES  |      | NULL    |       |
| Wait_time     | double unsigned     | YES  |      | NULL    |       |
| Backoff_time  | double unsigned     | YES  |      | NULL    |       |
| Request_count | bigint(20) unsigned | YES  |      | NULL    |       |
| Total_keys    | bigint(20) unsigned | YES  |      | NULL    |       |
| Process_keys  | bigint(20) unsigned | YES  |      | NULL    |       |
| DB            | varchar(64)         | YES  |      | NULL    |       |
| Index_ids     | varchar(100)        | YES  |      | NULL    |       |
| Is_internal   | tinyint(1) unsigned | YES  |      | NULL    |       |
| Digest        | varchar(64)         | YES  |      | NULL    |       |
| Stats         | varchar(512)        | YES  |      | NULL    |       |
| Cop_proc_avg  | double unsigned     | YES  |      | NULL    |       |
| Cop_proc_p90  | double unsigned     | YES  |      | NULL    |       |
| Cop_proc_max  | double unsigned     | YES  |      | NULL    |       |
| Cop_proc_addr | varchar(64)         | YES  |      | NULL    |       |
| Cop_wait_avg  | double unsigned     | YES  |      | NULL    |       |
| Cop_wait_p90  | double unsigned     | YES  |      | NULL    |       |
| Cop_wait_max  | double unsigned     | YES  |      | NULL    |       |
```

```
| Cop_wait_addr | varchar(64)     | YES |     | NULL   |       |
| Mem_max       | bigint(20) unsigned | YES |   | NULL   |       |
| Succ          | tinyint(1) unsigned | YES |   | NULL   |       |
| Query         | longblob unsigned | YES |     | NULL   |       |
+---------------+---------------------+------+-----+---------+-------+
```

4.5.2.2.1 STATISTICS table

The `STATISTICS` table provides information about table indexes:

```
desc statistics;
```

```
+--------------|---------------------|------|------|---------|-------+
| Field        | Type                | Null | Key | Default | Extra |
+--------------|---------------------|------|------|---------|-------+
| TABLE_CATALOG | varchar(512)    | YES |     | NULL   |       |
| TABLE_SCHEMA | varchar(64)      | YES |     | NULL   |       |
| TABLE_NAME   | varchar(64)      | YES |     | NULL   |       |
| NON_UNIQUE   | varchar(1)       | YES |     | NULL   |       |
| INDEX_SCHEMA | varchar(64)      | YES |     | NULL   |       |
| INDEX_NAME   | varchar(64)      | YES |     | NULL   |       |
| SEQ_IN_INDEX | bigint(2) UNSIGNED | YES |   | NULL   |       |
| COLUMN_NAME | varchar(21)       | YES |     | NULL   |       |
| COLLATION    | varchar(1)       | YES |     | NULL   |       |
| CARDINALITY | bigint(21) UNSIGNED | YES |   | NULL   |       |
| SUB_PART     | bigint(3) UNSIGNED | YES |    | NULL   |       |
| PACKED       | varchar(10)      | YES |     | NULL   |       |
| NULLABLE     | varchar(3)       | YES |     | NULL   |       |
| INDEX_TYPE   | varchar(16)      | YES |     | NULL   |       |
| COMMENT      | varchar(16)      | YES |     | NULL   |       |
| INDEX_COMMENT | varchar(1024)   | YES |     | NULL   |       |
+--------------|---------------------|------|------|---------|-------+
```

The following statements are equivalent:

```
SELECT * FROM INFORMATION_SCHEMA.STATISTICS
  WHERE table_name = 'tbl_name'
  AND table_schema = 'db_name'

SHOW INDEX
  FROM tbl_name
  FROM db_name
```

4.5.2.2.2 TABLES table

The `TABLES` table provides information about tables in databases:

```
SELECT * FROM tables WHERE table_schema='mysql' AND table_name='user';
```

```
*************************** 1. row ***************************
  TABLE_CATALOG: def
   TABLE_SCHEMA: mysql
     TABLE_NAME: user
     TABLE_TYPE: BASE TABLE
         ENGINE: InnoDB
        VERSION: 10
     ROW_FORMAT: Compact
     TABLE_ROWS: 0
 AVG_ROW_LENGTH: 0
    DATA_LENGTH: 0
MAX_DATA_LENGTH: 0
   INDEX_LENGTH: 0
      DATA_FREE: 0
 AUTO_INCREMENT: 0
    CREATE_TIME: 2019-03-29 09:17:27
    UPDATE_TIME: NULL
     CHECK_TIME: NULL
TABLE_COLLATION: utf8mb4_bin
       CHECKSUM: NULL
 CREATE_OPTIONS:
  TABLE_COMMENT:
  TIDB_TABLE_ID: 5
1 row in set (0.00 sec)
```

The following statements are equivalent:

```
SELECT table_name FROM INFORMATION_SCHEMA.TABLES
  WHERE table_schema = 'db_name'
  [AND table_name LIKE 'wild']

SHOW TABLES
  FROM db_name
  [LIKE 'wild']
```

4.5.2.2.3 TABLE_CONSTRAINTS table

The TABLE_CONSTRAINTS table describes which tables have constraints:

```
SELECT * FROM table_constraints WHERE constraint_type='UNIQUE';
```

```
*************************** 1. row ***************************
CONSTRAINT_CATALOG: def
```

```
 CONSTRAINT_SCHEMA: mysql
   CONSTRAINT_NAME: name
      TABLE_SCHEMA: mysql
        TABLE_NAME: help_topic
   CONSTRAINT_TYPE: UNIQUE
*************************** 2. row ***************************
CONSTRAINT_CATALOG: def
 CONSTRAINT_SCHEMA: mysql
   CONSTRAINT_NAME: tbl
      TABLE_SCHEMA: mysql
        TABLE_NAME: stats_meta
   CONSTRAINT_TYPE: UNIQUE
*************************** 3. row ***************************
CONSTRAINT_CATALOG: def
 CONSTRAINT_SCHEMA: mysql
   CONSTRAINT_NAME: tbl
      TABLE_SCHEMA: mysql
        TABLE_NAME: stats_histograms
   CONSTRAINT_TYPE: UNIQUE
*************************** 4. row ***************************
CONSTRAINT_CATALOG: def
 CONSTRAINT_SCHEMA: mysql
   CONSTRAINT_NAME: tbl
      TABLE_SCHEMA: mysql
        TABLE_NAME: stats_buckets
   CONSTRAINT_TYPE: UNIQUE
*************************** 5. row ***************************
CONSTRAINT_CATALOG: def
 CONSTRAINT_SCHEMA: mysql
   CONSTRAINT_NAME: delete_range_index
      TABLE_SCHEMA: mysql
        TABLE_NAME: gc_delete_range
   CONSTRAINT_TYPE: UNIQUE
*************************** 6. row ***************************
CONSTRAINT_CATALOG: def
 CONSTRAINT_SCHEMA: mysql
   CONSTRAINT_NAME: delete_range_done_index
      TABLE_SCHEMA: mysql
        TABLE_NAME: gc_delete_range_done
   CONSTRAINT_TYPE: UNIQUE
6 rows in set (0.00 sec)
```

- The `CONSTRAINT_TYPE` value can be `UNIQUE`, `PRIMARY KEY`, or `FOREIGN KEY`.
- The `UNIQUE` and `PRIMARY KEY` information is similar to the result of the `SHOW INDEX`

statement.

4.5.2.2.4 TIDB_HOT_REGIONS table

The TIDB_HOT_REGIONS table provides information about hot spot Regions.

```
desc TIDB_HOT_REGIONS;
```

```
+---------------+---------------------+------+-----+---------+-------+
| Field         | Type                | Null | Key | Default | Extra |
+---------------+---------------------+------+-----+---------+-------+
| TABLE_ID      | bigint(21) unsigned | YES  |     | <null>  |       |
| INDEX_ID      | bigint(21) unsigned | YES  |     | <null>  |       |
| DB_NAME       | varchar(64)         | YES  |     | <null>  |       |
| TABLE_NAME    | varchar(64)         | YES  |     | <null>  |       |
| INDEX_NAME    | varchar(64)         | YES  |     | <null>  |       |
| TYPE          | varchar(64)         | YES  |     | <null>  |       |
| MAX_HOT_DEGREE | bigint(21) unsigned | YES  |     | <null>  |       |
| REGION_COUNT  | bigint(21) unsigned | YES  |     | <null>  |       |
| FLOW_BYTES    | bigint(21) unsigned | YES  |     | <null>  |       |
+---------------+---------------------+------+-----+---------+-------+
```

4.5.2.2.5 TIDB_INDEXES table

The TIDB_INDEXES table provides the INDEX information of all tables.

```
desc TIDB_INDEXES;
```

```
+--------------+---------------------+------+-----+---------+-------+
| Field        | Type                | Null | Key | Default | Extra |
+--------------+---------------------+------+-----+---------+-------+
| TABLE_SCHEMA | varchar(64)         | YES  |     | <null>  |       |
| TABLE_NAME   | varchar(64)         | YES  |     | <null>  |       |
| NON_UNIQUE   | bigint(21) unsigned | YES  |     | <null>  |       |
| KEY_NAME     | varchar(64)         | YES  |     | <null>  |       |
| SEQ_IN_INDEX | bigint(21) unsigned | YES  |     | <null>  |       |
| COLUMN_NAME  | varchar(64)         | YES  |     | <null>  |       |
| SUB_PART     | bigint(21) unsigned | YES  |     | <null>  |       |
| INDEX_COMMENT | varchar(2048)      | YES  |     | <null>  |       |
| INDEX_ID     | bigint(21) unsigned | YES  |     | <null>  |       |
+--------------+---------------------+------+-----+---------+-------+
```

Fields in the TIDB_INDEXES table are described as follows:

- TABLE_SCHEMA: The name of the schema to which the index belongs.

- `TABLE_NAME`: The name of the table to which the index belongs.
- `NON_UNIQUE`: If the index is unique, the value is `0`; otherwise, the value is `1`.
- `KEY_NAME`: The index name. If the index is the primary key, the name is `PRIMARY`.
- `SEQ_IN_INDEX`: The sequential number of columns in the index, which starts from `1`.
- `COLUMN_NAME`: The name of the column where the index is located.
- `SUB_PART`: The prefix length of the index. If the the column is partly indexed, the `SUB_PART` value is the count of the indexed characters; otherwise, the value is `NULL`.
- `INDEX_COMMENT`: The comment of the index, which is made when the index is created.
- `INDEX_ID`: The index ID.

4.5.2.2.6 TIKV_REGION_PEERS table

The `TIKV_REGION_PEERS` table provides the peer information of all Regions.

```
desc TIKV_REGION_PEERS;
```

```
+--------------+---------------------+------+-----+---------+-------+
| Field        | Type                | Null | Key | Default | Extra |
+--------------+---------------------+------+-----+---------+-------+
| REGION_ID    | bigint(21) unsigned | YES  |     | <null>  |       |
| PEER_ID      | bigint(21) unsigned | YES  |     | <null>  |       |
| STORE_ID     | bigint(21) unsigned | YES  |     | <null>  |       |
| IS_LEARNER   | tinyint(1) unsigned | YES  |     | <null>  |       |
| IS_LEADER    | tinyint(1) unsigned | YES  |     | <null>  |       |
| STATUS       | varchar(10)         | YES  |     | <null>  |       |
| DOWN_SECONDS | bigint(21) unsigned | YES  |     | <null>  |       |
+--------------+---------------------+------+-----+---------+-------+
```

4.5.2.2.7 TIKV_REGION_STATUS table

The `TIKV_REGION_STATUS` table provides the status information of all Regions.

```
desc TIKV_REGION_STATUS;
```

```
+------------------+---------------------+------+-----+---------+-------+
| Field            | Type                | Null | Key | Default | Extra |
+------------------+---------------------+------+-----+---------+-------+
| REGION_ID        | bigint(21) unsigned | YES  |     | <null>  |       |
| START_KEY        | text                | YES  |     | <null>  |       |
| END_KEY          | text                | YES  |     | <null>  |       |
| EPOCH_CONF_VER   | bigint(21) unsigned | YES  |     | <null>  |       |
| EPOCH_VERSION    | bigint(21) unsigned | YES  |     | <null>  |       |
| WRITTEN_BYTES    | bigint(21) unsigned | YES  |     | <null>  |       |
| READ_BYTES       | bigint(21) unsigned | YES  |     | <null>  |       |
| APPROXIMATE_SIZE | bigint(21) unsigned | YES  |     | <null>  |       |
| APPROXIMATE_KEYS | bigint(21) unsigned | YES  |     | <null>  |       |
```

```
+-----------------+--------------------+------+-----+---------+-------+
```

4.5.2.2.8 TIKV_STORE_STATUS table

The `TIKV_STORE_STATUS` table provides the status information of all TiKV Stores.

```
desc TIKV_STORE_STATUS;
```

```
+------------------+--------------------+------+-----+---------+-------+
| Field            | Type               | Null | Key | Default | Extra |
+------------------+--------------------+------+-----+---------+-------+
| STORE_ID         | bigint(21) unsigned | YES |  | <null> |       |
| ADDRESS          | varchar(64)        | YES |     | <null> |       |
| STORE_STATE      | bigint(21) unsigned | YES |  | <null> |       |
| STORE_STATE_NAME | varchar(64)        | YES |     | <null> |       |
| LABEL            | json unsigned      | YES |     | <null> |       |
| VERSION          | varchar(64)        | YES |     | <null> |       |
| CAPACITY         | varchar(64)        | YES |     | <null> |       |
| AVAILABLE        | varchar(64)        | YES |     | <null> |       |
| LEADER_COUNT     | bigint(21) unsigned | YES |  | <null> |       |
| LEADER_WEIGHT    | bigint(21) unsigned | YES |  | <null> |       |
| LEADER_SCORE     | bigint(21) unsigned | YES |  | <null> |       |
| LEADER_SIZE      | bigint(21) unsigned | YES |  | <null> |       |
| REGION_COUNT     | bigint(21) unsigned | YES |  | <null> |       |
| REGION_WEIGHT    | bigint(21) unsigned | YES |  | <null> |       |
| REGION_SCORE     | bigint(21) unsigned | YES |  | <null> |       |
| REGION_SIZE      | bigint(21) unsigned | YES |  | <null> |       |
| START_TS         | datetime unsigned | YES |    | <null> |       |
| LAST_HEARTBEAT_TS | datetime unsigned | YES | | <null> |       |
| UPTIME           | varchar(64)        | YES |     | <null> |       |
+------------------+--------------------+------+-----+---------+-------+
```

4.5.2.2.9 USER_PRIVILEGES table

The `USER_PRIVILEGES` table provides information about global privileges. This information comes from the `mysql.user` system table:

```
desc USER_PRIVILEGES;
```

```
+---------------/-------------/------/------/---------/-------+
| Field        | Type        | Null | Key | Default | Extra |
+---------------/-------------/------/------/---------/-------+
| GRANTEE      | varchar(81) | YES |     | NULL   |       |
| TABLE_CATALOG | varchar(512) | YES |  | NULL   |       |
| PRIVILEGE_TYPE | varchar(64) | YES |  | NULL   |       |
```

```
| IS_GRANTABLE | varchar(3) | YES |     | NULL   |      |
+---------------/--------------/------/------/---------/-------+
4 rows in set (0.00 sec)
```

Fields in the `USER_PRIVILEGES` table are described as follows:

- `GRANTEE`: The name of the granted user, which is in the format of `'user_name'@'host_name'`.
- `TABLE_CATALOG`: The name of the catalog to which the table belongs. This value is always `def`.
- `PRIVILEGE_TYPE`: The privilege type to be granted. Only one privilege type is shown in each row.
- `IS_GRANTABLE`: If you have the `GRANT OPTION` privilege, the value is `YES`; otherwise, the value is `NO`.

4.5.2.2.10 VIEWS table

The `VIEWS` table provides information about SQL views:

```
create view test.v1 as select 1;
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
select * from views;
```

```
*************************** 1. row ***************************
       TABLE_CATALOG: def
        TABLE_SCHEMA: test
          TABLE_NAME: v1
     VIEW_DEFINITION: select 1
        CHECK_OPTION: CASCADED
        IS_UPDATABLE: NO
             DEFINER: root@127.0.0.1
       SECURITY_TYPE: DEFINER
CHARACTER_SET_CLIENT: utf8
COLLATION_CONNECTION: utf8_general_ci
1 row in set (0.00 sec)
```

Fields in the `VIEWS` table are described as follows:

- `TABLE_CATALOG`: The name of the catalog to which the view belongs. This value is always `def`.
- `TABLE_SCHEMA`: The name of the schema to which the view belongs.
- `TABLE_NAME`: The view name.

- VIEW_DEFINITION: The definition of view, which is made by the SELECT statement when the view is created.
- CHECK_OPTION: The CHECK_OPTION value. The value options are NONE, CASCADE, and LOCAL.
- IS_UPDATABLE: Whether UPDATE/INSERT/DELETE is applicable to the view. In TiDB, the value is always NO.
- DEFINER: The name of the user who creates the view, which is in the format of 'user_name'@'host_name'.
- SECURITY_TYPE: The value of SQL SECURITY. The value options are DEFINER and INVOKER.
- CHARACTER_SET_CLIENT: The value of the character_set_client session variable when the view is created.
- COLLATION_CONNECTION: The value of the collation_connection session variable when the view is created.

4.5.2.3 TIDB_INDEXES table

The TIDB_INDEXES table provides index-related information.

```
desc tidb_indexes\G
```

```
*************************** 1. row ***************************
      Table: TIDB_INDEXES
Create Table: CREATE TABLE `TIDB_INDEXES` (
  `TABLE_SCHEMA` varchar(64) DEFAULT NULL,
  `TABLE_NAME` varchar(64) DEFAULT NULL,
  `NON_UNIQUE` bigint(21) unsigned DEFAULT NULL,
  `KEY_NAME` varchar(64) DEFAULT NULL,
  `SEQ_IN_INDEX` bigint(21) unsigned DEFAULT NULL,
  `COLUMN_NAME` varchar(64) DEFAULT NULL,
  `SUB_PART` bigint(21) unsigned DEFAULT NULL,
  `INDEX_COMMENT` varchar(2048) DEFAULT NULL,
  `INDEX_ID` bigint(21) unsigned DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
1 row in set (0.00 sec)
```

INDEX_ID is the unique ID that TiDB allocates for each index. It can be used to do a join operation with INDEX_ID obtained from another table or API.

For example, you can obtain TABLE_ID and INDEX_ID that are involved in some slow query in the SLOW_QUERY table and then obtain the specific index information using the following SQL statements:

```
select
   tidb_indexes.*
from
   tidb_indexes,
```

```
   tables
where
   tidb_indexes.table_schema = tables.table_schema
   and tidb_indexes.table_name = tidb_indexes.table_name
   and tables.tidb_table_id = ?
   and index_id = ?
```

4.5.2.4 TIDB_HOT_REGIONS table

The TIDB_HOT_REGIONS table provides the hot Region information in the current TiKV instance.

```
desc tidb_hot_regions\G
```

```
*************************** 1. row ***************************
      Table: TIDB_HOT_REGIONS
Create Table: CREATE TABLE `TIDB_HOT_REGIONS` (
  `TABLE_ID` bigint(21) unsigned DEFAULT NULL,
  `INDEX_ID` bigint(21) unsigned DEFAULT NULL,
  `DB_NAME` varchar(64) DEFAULT NULL,
  `TABLE_NAME` varchar(64) DEFAULT NULL,
  `INDEX_NAME` varchar(64) DEFAULT NULL,
  `TYPE` varchar(64) DEFAULT NULL,
  `MAX_HOT_DEGREE` bigint(21) unsigned DEFAULT NULL,
  `REGION_COUNT` bigint(21) unsigned DEFAULT NULL,
  `FLOW_BYTES` bigint(21) unsigned DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
1 row in set (0.00 sec)
```

- TABLE_ID and INDEX_ID are IDs generated for the corresponding table and index in TiDB.
- TYPE is the type for a hot Region. Its value can be READ or WRITE.

4.5.2.5 TIKV_STORE_STATUS table

The TIKV_STORE_STATUS table shows some basic information of TiKV nodes via PD's API, like the ID allocated in the cluster, address and port, and status, capacity, and the number of Region leaders of the current node.

```
desc tikv_store_status\G
```

```
*************************** 1. row ***************************
      Table: TIKV_STORE_STATUS
Create Table: CREATE TABLE `TIKV_STORE_STATUS` (
  `STORE_ID` bigint(21) unsigned DEFAULT NULL,
```

```
  `ADDRESS` varchar(64) DEFAULT NULL,
  `STORE_STATE` bigint(21) unsigned DEFAULT NULL,
  `STORE_STATE_NAME` varchar(64) DEFAULT NULL,
  `LABEL` json unsigned DEFAULT NULL,
  `VERSION` varchar(64) DEFAULT NULL,
  `CAPACITY` varchar(64) DEFAULT NULL,
  `AVAILABLE` varchar(64) DEFAULT NULL,
  `LEADER_COUNT` bigint(21) unsigned DEFAULT NULL,
  `LEADER_WEIGHT` bigint(21) unsigned DEFAULT NULL,
  `LEADER_SCORE` bigint(21) unsigned DEFAULT NULL,
  `LEADER_SIZE` bigint(21) unsigned DEFAULT NULL,
  `REGION_COUNT` bigint(21) unsigned DEFAULT NULL,
  `REGION_WEIGHT` bigint(21) unsigned DEFAULT NULL,
  `REGION_SCORE` bigint(21) unsigned DEFAULT NULL,
  `REGION_SIZE` bigint(21) unsigned DEFAULT NULL,
  `START_TS` datetime unsigned DEFAULT NULL,
  `LAST_HEARTBEAT_TS` datetime unsigned DEFAULT NULL,
  `UPTIME` varchar(64) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
1 row in set (0.01 sec)
```

4.5.2.6 TIKV_REGION_STATUS table

The TIKV_REGION_STATUS table shows some basic information of TiKV Regions via PD's API, like the Region ID, starting and ending key-values, and read and write traffic.

```
desc tikv_region_status\G
```

```
*************************** 1. row ***************************
      Table: TIKV_REGION_STATUS
Create Table: CREATE TABLE `TIKV_REGION_STATUS` (
  `REGION_ID` bigint(21) unsigned DEFAULT NULL,
  `START_KEY` text DEFAULT NULL,
  `END_KEY` text DEFAULT NULL,
  `EPOCH_CONF_VER` bigint(21) unsigned DEFAULT NULL,
  `EPOCH_VERSION` bigint(21) unsigned DEFAULT NULL,
  `WRITTEN_BYTES` bigint(21) unsigned DEFAULT NULL,
  `READ_BYTES` bigint(21) unsigned DEFAULT NULL,
  `APPROXIMATE_SIZE` bigint(21) unsigned DEFAULT NULL,
  `APPROXIMATE_KEYS` bigint(21) unsigned DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
1 row in set (0.00 sec)
```

You can implement the `top confver`, `top read` and `top write` operations in pd-ctl via the `ORDER BY X LIMIT Y` operation on the `EPOCH_CONF_VER`, `WRITTEN_BYTES` and

READ_BYTES columns.

You can query the top 3 Regions with the most write data using the following SQL statement:

```
select * from tikv_region_status order by written_bytes desc limit 3;
```

4.5.2.7 TIKV_REGION_PEERS table

The TIKV_REGION_PEERS table shows detailed information of a single Region node in TiKV, like whether it is a learner or leader.

```
desc tikv_region_peers\G
```

```
*************************** 1. row ***************************
      Table: TIKV_REGION_PEERS
Create Table: CREATE TABLE `TIKV_REGION_PEERS` (
  `REGION_ID` bigint(21) unsigned DEFAULT NULL,
  `PEER_ID` bigint(21) unsigned DEFAULT NULL,
  `STORE_ID` bigint(21) unsigned DEFAULT NULL,
  `IS_LEARNER` tinyint(1) unsigned DEFAULT NULL,
  `IS_LEADER` tinyint(1) unsigned DEFAULT NULL,
  `STATUS` varchar(10) DEFAULT NULL,
  `DOWN_SECONDS` bigint(21) unsigned DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
1 row in set (0.00 sec)
```

For example, you can query the specific TiKV addresses for the top 3 Regions with the maximum value of WRITTEN_BYTES using the following SQL statement:

```
select
   address,
   tikv.address,
   region.region_id,
from
   tikv_store_status tikv,
   tikv_region_peers peer,
   (
      select
         *
      from
         tikv_region_status region
      order by
         written_bytes desc limit 3
   )
   region
where
```

```
  region.region_id = peer.region_id
  and peer.is_leader = 1
  and peer.store_id = tikv.region_id
```

4.5.2.8 ANALYZE_STATUS table

The ANALYZE_STATUS table shows the execution status of the ANALYZE command in the current cluster.

```
desc analyze_status\G
```

```
*************************** 1. row ***************************
      Table: ANALYZE_STATUS
Create Table: CREATE TABLE `ANALYZE_STATUS` (
  `TABLE_SCHEMA` varchar(64) DEFAULT NULL,
  `TABLE_NAME` varchar(64) DEFAULT NULL,
  `PARTITION_NAME` varchar(64) DEFAULT NULL,
  `JOB_INFO` varchar(64) DEFAULT NULL,
  `PROCESSED_ROWS` bigint(20) unsigned DEFAULT NULL,
  `START_TIME` datetime unsigned DEFAULT NULL,
  `STATE` varchar(64) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
1 row in set (0.00 sec)
```

The STATE column shows the execution status of a specific ANALYZE task. Its value can be pending, running,finished or failed.

4.5.2.9 SLOW_QUERY table

The SLOW_QUERY table maps slow query logs. Its column names and field names of slow query logs have an one-to-one corresponse relationship. For details, see Identify Slow Queries.

```
desc slow_query\G
```

```
*************************** 1. row ***************************
      Table: SLOW_QUERY
Create Table: CREATE TABLE `SLOW_QUERY` (
  `Time` timestamp unsigned NULL DEFAULT NULL,
  `Txn_start_ts` bigint(20) unsigned DEFAULT NULL,
  `User` varchar(64) DEFAULT NULL,
  `Host` varchar(64) DEFAULT NULL,
  `Conn_ID` bigint(20) unsigned DEFAULT NULL,
  `Query_time` double unsigned DEFAULT NULL,
  `Process_time` double unsigned DEFAULT NULL,
```

```
  `Wait_time` double unsigned DEFAULT NULL,
  `Backoff_time` double unsigned DEFAULT NULL,
  `Request_count` bigint(20) unsigned DEFAULT NULL,
  `Total_keys` bigint(20) unsigned DEFAULT NULL,
  `Process_keys` bigint(20) unsigned DEFAULT NULL,
  `DB` varchar(64) DEFAULT NULL,
  `Index_ids` varchar(100) DEFAULT NULL,
  `Is_internal` tinyint(1) unsigned DEFAULT NULL,
  `Digest` varchar(64) DEFAULT NULL,
  `Stats` varchar(512) DEFAULT NULL,
  `Cop_proc_avg` double unsigned DEFAULT NULL,
  `Cop_proc_p90` double unsigned DEFAULT NULL,
  `Cop_proc_max` double unsigned DEFAULT NULL,
  `Cop_proc_addr` varchar(64) DEFAULT NULL,
  `Cop_wait_avg` double unsigned DEFAULT NULL,
  `Cop_wait_p90` double unsigned DEFAULT NULL,
  `Cop_wait_max` double unsigned DEFAULT NULL,
  `Cop_wait_addr` varchar(64) DEFAULT NULL,
  `Mem_max` bigint(20) unsigned DEFAULT NULL,
  `Query` varchar(4096) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_bin
1 row in set (0.00 sec)
```

4.5.2.10 Unsupported Information Schema Tables

The following `INFORMATION_SCHEMA` tables are present in TiDB, but will always return zero rows:

- COLUMN_PRIVILEGES
- EVENTS
- FILES
- GLOBAL_STATUS
- GLOBAL_VARIABLES
- OPTIMIZER_TRACE
- PARAMETERS
- PARTITIONS
- PLUGINS
- PROFILING
- REFERENTIAL_CONSTRAINTS
- ROUTINES
- SCHEMA_PRIVILEGES
- SESSION_STATUS
- TABLESPACES
- TABLE_PRIVILEGES
- TRIGGERS

4.6 Error Codes and Troubleshooting

This document describes the problems encountered during the use of TiDB and provides the solutions.

4.6.1 Error codes

TiDB is compatible with the error codes in MySQL, and in most cases returns the same error code as MySQL. In addition, TiDB has the following unique error codes:

Error code	Description	Solution
8001	The memory used by the request exceeds the threshold limit for the TiDB memory usage.	Increase the value of the system variable with the `tidb_mem_quota` ↪ prefix.

Error code	Description	Solution
8002	To guarantee consistency, a transaction with the `SELECT FOR UPDATE` statement cannot be retried when it encounters a commit conflict. TiDB rolls back the transaction and returns this error.	Retry the failed transaction.

Error code	Description	Solution
8003	If the data in a row is not consistent with the index when executing the `ADMIN CHECK TABLE` command, TiDB returns this error.	
8004	A single transaction is too large.	See the answer of the error message `transaction too large` for the cause and solution.
8005	Transactions in TiDB encounter write conflicts.	See the Troubleshoot section for the cause and solution.

Error code	Description	Solution
9001	The PD request timed out.	Check the state/monitor/log of the PD server and the network between the TiDB server and the PD server.
9002	The TiKV request timed out.	Check the state/monitor/log of the TiKV server and the network between the TiDB server and the TiKV server.
9003	The TiKV server is busy and this usually occurs when the workload is too high.	Check the state/monitor/log of the TiKV server.

Error code	Description	Solution
9004	This error occurs when a large number of transactional conflicts exist in the database.	Check the code of application.
9005	A certain Raft Group is not available, such as the number of replicas is not enough. This error usually occurs when the TiKV server is busy or the TiKV node is down.	Check the state/monitor/log of the TiKV server.

Error code	Description	Solution
9006	The interval of GC Life Time is too short and the data that should be read by the long transactions might be cleared.	Extend the interval of GC Life Time.
9500	A single transaction is too large.	See the error message `transaction too large` for the solution.
9007	Transactions in TiKV encounter write conflicts.	See the Troubleshoot section for the cause and solution.

4.6.2 Troubleshooting

See the troubleshooting and FAQ documents.

4.7 Connectors and APIs

Database Connectors provide connectivity to the TiDB server for client programs. APIs provide low-level access to the MySQL protocol and MySQL resources. Both Connectors and the APIs enable you to connect and execute MySQL statements from another language or environment, including ODBC, Java (JDBC), Perl, Python, PHP, Ruby and C.

TiDB is compatible with all Connectors and APIs of MySQL (5.6, 5.7), including:

- MySQL Connector/C++
- MySQL Connector/J
- MySQL Connector/Net
- MySQL Connector/ODBC
- MySQL Connector/Python
- MySQL C API
- MySQL PHP API
- MySQL Perl API
- MySQL Python API
- MySQL Ruby APIs
- MySQL Tcl API
- MySQL Eiffel Wrapper
- Mysql Go API

4.7.1 Connect to TiDB using MySQL Connectors

Oracle develops the following APIs and TiDB is compatible with all of them:

Note:

- To connect to TiDB using a MySQL Connector from MySQL 8.0, you must explicitly specify `default-auth=mysql_native_password`, because `mysql_native_password` is no longer the default plugin.

- MySQL Connector/C++: to enable C++ applications to connect to MySQL
- MySQL Connector/J: to enable Java applications to connect to MySQL using the standard JDBC API
- MySQL Connector/Net: to enable .Net applications to connect to MySQL; MySQL for Visual Studio uses this; support Microsoft Visual Studio 2012, 2013, 2015 and 2017 versions
- MySQL Connector/ODBC: the standard ODBC API; support Windows, Unix, and OS X platforms
- MySQL Connector/Python: to enable Python applications to connect to MySQL, compliant with the Python DB API version 2.0

4.7.2 Connect to TiDB using MySQL C API

If you use C language programs to connect to TiDB, you can connect to `libmysqlclient` directly and use the MySQL C API. This is one of the major connection methods using C language, widely used by various clients and APIs, including Connector/C.

4.7.3 Connect to TiDB using third-party MySQL APIs

The third-party APIs are not developed by Oracle. The following table lists the commonly used third-party APIs:

Environment	API	Type	Notes
Ada	GNU Ada MySQL Bindings	`libmysqlclient`	See MySQL Bindings for GNU Ada
C	C API	`libmysqlclient`	See MySQL C API
C++	Connector/C++	`libmysqlclient`	See MySQL Connector/C++ Developer Guide
	MySQL++	`libmysqlclient`	See MySQL++ Web site
	MySQL wrapped	`libmysqlclient`	See MySQL wrapped
Go	go-sql-driver	Native Driver	See Mysql Go API
Cocoa	MySQL-Cocoa	`libmysqlclient`	Compatible with the Objective-C Cocoa environment. See http://mysql-cocoa.sourceforge.net/
D	MySQL for D	`libmysqlclient`	See MySQL for D
Eiffel	Eiffel MySQL	`libmysqlclient`	See Section 27.14, "MySQL Eiffel Wrapper"
Erlang	`erlang-mysql-driver`	`libmysqlclient`	See `erlang-mysql-driver`
Haskell	Haskell MySQL Bindings	Native Driver	See Brian O'Sullivan's pure Haskell MySQL bindings
	`hsql-mysql`	`libmysqlclient`	See MySQL driver for Haskell
Java/JDBC	Connector/J	Native Driver	See MySQL Connector/J 5.1 Developer Guide
Lua	LuaSQL	`libmysqlclient`	See LuaSQL
.NET/Mono	Connector/Net	Native Driver	See MySQL Connector/Net Developer Guide
Objective Caml	OBjective Caml MySQL Bindings	`libmysqlclient`	See MySQL Bindings for Objective Caml

Environment	API	Type	Notes
Octave	Database bindings for GNU Octave	`libmysqlclient`	See Database bindings for GNU Octave
ODBC	Connector/ODBC	`libmysqlclient`	See MySQL Connector/ODBC Developer Guide
Perl	`DBI/DBD::mysql`	`libmysqlclient`	See Section 27.10, "MySQL Perl API"
	`Net::MySQL`	Native Driver	See `Net::MySQL` at CPAN
PHP	`mysql`, `ext/mysql`interface (deprecated)	`libmysqlclient`	See Original MySQL API
	`mysqli`, `ext/mysqli`interface	`libmysqlclient`	See MySQL Improved Extension
	`PDO_MYSQL`	`libmysqlclient`	See MySQL Functions (PDO_MYSQL)
	PDO mysqlnd	Native Driver	
Python	Connector/Python	Native Driver	See MySQL Connector/Python Developer Guide
Python	Connector/Python C Extension	`libmysqlclient`	See MySQL Connector/Python Developer Guide
	MySQLdb	`libmysqlclient`	See Section 27.11, "MySQL Python API"
Ruby	MySQL/Ruby	`libmysqlclient`	Uses `libmysqlclient`. See Section 27.12.1, "The MySQL/Ruby API"
	Ruby/MySQL	Native Driver	See Section 27.12.2, "The Ruby/MySQL API"
Scheme	`Myscsh`	`libmysqlclient`	See `Myscsh`
SPL	`sql_mysql`	`libmysqlclient`	See `sql_mysql` for SPL
Tcl	MySQLtcl	`libmysqlclient`	See Section 27.13, "MySQL Tcl API"

4.7.4 Connector versions supported by TiDB

Connector	Connector Version
Connector/C	6.1.0 GA
Connector/C++	1.0.5 GA
Connector/J	5.1.8
Connector/Net	6.9.9 GA

Connector	Connector Version
Connector/Net	6.8.8 GA
Connector/ODBC	5.1
Connector/ODBC	3.51 (Unicode not supported)
Connector/Python	2.0
Connector/Python	1.2

4.8 Garbage Collection (GC)

4.8.1 GC Overview

TiDB uses MVCC to control transaction concurrency. When you update the data, the original data is not deleted immediately but is kept together with the new data, with a timestamp to distinguish the version. The goal of Garbage Collection (GC) is to clear the obsolete data.

4.8.1.1 GC process

Each TiDB cluster contains a TiDB instance that is selected as the GC leader, which controls the GC process.

GC runs periodically on TiDB. The default frequency is once every 10 minutes. For each GC, TiDB firstly calculates a timestamp called "safe point" (defaults to the current time minus 10 minutes). Then, TiDB clears the obsolete data under the premise that all the snapshots after the safe point retain the integrity of the data. Specifically, there are three steps involved in the GC process:

1. Resolve Locks
2. Delete Ranges
3. Do GC

4.8.1.1.1 Resolve Locks

The TiDB transaction model is implemented based on Google's Percolator. It's mainly a two-phase commit protocol with some practical optimizations. When the first phase is finished, all the related keys are locked. Among these locks, one is the primary lock and the others are secondary locks which contain a pointer to the primary lock; in the second phase, the key with the primary lock gets a write record and its lock is removed. The write record indicates the write or delete operation in the history or the transactional rollback record of this key. The type of write record that replaces the primary lock indicates whether the corresponding transaction is committed successfully. Then all the secondary locks are replaced successively. If the threads fail to replace the secondary locks, these locks are retained.

The Resolve Locks step rolls back or commits the locks before the safe point, depending on whether their primary key has been committed or not. If the primary key is also retained, the transaction times out and is rolled back. This step is required. Once GC has cleared the write record of the primary lock, you can never know whether this transaction is successful or not. Also, if the the transaction contains retained secondary keys, it's important to know whether it should be rolled back or committed. As a result, data consistency cannot be guaranteed.

In the Resolve Lock step, the GC leader processes requests from all Regions. From TiDB 3.0, this process runs concurrently by default, with the default concurrency equal to the number of TiKV nodes in the cluster. For more details on how to configure, see GC Configuration.

4.8.1.1.2 Delete Ranges

A great amount of data with consecutive keys is removed during operations such as `DROP TABLE/INDEX`. Removing each key and performing GC later for them can result in low execution efficiency on storage reclaiming. In such scenarios, TiDB actually does not delete each key. Instead, it only records the range to be removed and the timestamp of the deletion. Then the Delete Ranges step performs a fast physical deletion on the ranges whose timestamp is before the safe point.

4.8.1.1.3 Do GC

The Do GC step clears the outdated versions for all keys. To guarantee that all timestamps after the safe point have consistent snapshots, this step deletes the data committed before the safe point, but retains the last write before the safe point as long as it is not a deletion.

In the previous GC mechanism for TiDB 2.1 and earlier versions, the GC leader sends GC requests to all Regions. From TiDB 3.0, the GC leader only uploads the safe point to PD for each TiKV node to obtain. When the TiKV node detects a change on the safe point, it performs GC on all leader Regions on the current node. In the meantime, the GC leader can trigger the next round of GC.

> **Note:**
>
> You can modify the `tikv_gc_mode` to use the previous GC mechanism. For more details, refer to GC Configuration.

4.8.2 GC Configuration

The GC (Garbage Collection) configuration and operational status are recorded in the `mysql.tidb` system table. You can use SQL statements to query or modify them:

```
mysql> select VARIABLE_NAME, VARIABLE_VALUE from mysql.tidb;
+-------------------------+----------------------------------------------------------------
   ↪
| VARIABLE_NAME         | VARIABLE_VALUE
   ↪
   ↪ |
+-------------------------+----------------------------------------------------------------
   ↪
| bootstrapped          | True
   ↪
   ↪ |
| tidb_server_version   | 33
   ↪
   ↪ |
| system_tz             | UTC
   ↪
   ↪ |
| tikv_gc_leader_uuid   | 5afd54a0ea40005
   ↪
   ↪ |
| tikv_gc_leader_desc   | host:tidb-cluster-tidb-0, pid:215, start at
   ↪ 2019-07-15 11:09:14.029668932 +0000 UTC m=+0.463731223 |
| tikv_gc_leader_lease  | 20190715-12:12:14 +0000
   ↪                                                                |
| tikv_gc_enable        | true
   ↪
   ↪ |
| tikv_gc_run_interval  | 10m0s
   ↪
   ↪ |
| tikv_gc_life_time     | 10m0s
   ↪
   ↪ |
| tikv_gc_last_run_time | 20190715-12:09:14 +0000
   ↪                                                               |
| tikv_gc_safe_point    | 20190715-11:59:14 +0000
   ↪                                                                |
| tikv_gc_auto_concurrency | true
   ↪
   ↪ |
| tikv_gc_mode          | distributed
   ↪
   ↪ |
+-------------------------+----------------------------------------------------------------
```

```
    ↪
13 rows in set (0.00 sec)
```

For example, the following statement makes GC keep history data for the most recent 24 hours:

```
update mysql.tidb set VARIABLE_VALUE="24h" where VARIABLE_NAME="
    ↪ tikv_gc_life_time";
```

Note:

In addition to the following GC configuration parameters, the `mysql.tidb` ↪ system table also contains records that store the status of the storage components in a TiDB cluster, among which GC related ones are included, as listed below:

- `tikv_gc_leader_uuid`, `tikv_gc_leader_desc` and `tikv_gc_leader_lease`: Records the information of the GC leader
- `tikv_gc_last_run_time`: The duration of the previous GC
- `tikv_gc_safe_point`: The safe point for the current GC

4.8.2.1 `tikv_gc_enable`

- Enables or disables GC
- Default: `true`

4.8.2.2 `tikv_gc_run_interval`

- Specifies the GC interval, in the format of Go Duration, for example, `"1h30m"`, and `"15m"`
- Default: `"10m0s"`

4.8.2.3 `tikv_gc_life_time`

- The time limit during which data is retained for each GC, in the format of Go Duration. When a GC happens, the current time minus this value is the safe point.
- Default: `"10m0s"`

Note:

The value of `tikv_gc_life_time` must be greater than that of `max-txn-time-use` in the TiDB configuration file by at least 10 seconds, and must than or equal to 10 minutes.

In scenarios of frequent updates, a large value (days or even months) for `tikv_gc_life_time` may cause potential issues, such as:

- Larger storage use
- A large amount of history data may affect performance to a certain degree, especially for range queries such as `select count(*)from t`

4.8.2.4 `tikv_gc_mode`

- Specifies the GC mode. Possible values are:
  - `"distributed"` (default): Distributed GC mode. In the Do GC step, the GC leader on the TiDB side uploads the safe point to PD. Each TiKV node obtains the safe point respectively and performs GC on all leader Regions on the current node. This mode is is supported from TiDB 3.0.
  - `"central"`: Central GC mode. In the Do GC step, the GC leader sends GC requests to all Regions. This mode is adopted by TiDB 2.1 or earlier versions.

4.8.2.5 `tikv_gc_auto_concurrency`

- Controls whether to let TiDB automatically specify the GC concurrency, or the maximum number of GC threads allowed concurrently.

  When `tikv_gc_mode` is set to `"distributed"`, GC concurrency works in the Resolve Locks step. When `tikv_gc_mode` is set to `"central"`, it is applied to both the Resolve Locks and Do GC steps.

  - `true`(default): Automatically use the number of TiKV nodes in the cluster as the GC concurrency
  - `false`: Use the value of `tikv_gc_concurrency` as the GC concurrency

4.8.2.6 `tikv_gc_concurrency`

- Specifies the GC concurrency manually. This parameter works only when you set `tikv_gc_auto_concurrency` to `false`.
- Default: 2

4.8.2.7 Notes on GC process changes

Since TiDB 3.0, some configuration options have changed with support for the distributed GC mode and concurrent Resolve Locks processing. The changes are shown in the following table:

Version/Configuration	Resolve Locks	Do GC
2.x	Serial	Concurrent
3.0 `tikv_gc_mode = centered` `tikv_gc_auto_concurrency = false`	Concurrent	Concurrent
3.0 `tikv_gc_mode = centered` `tikv_gc_auto_concurrency = true`	Auto-concurrent	Auto-concurrent
3.0 `tikv_gc_mode = distributed` `tikv_gc_auto_concurrency = false`	Concurrent	Distributed
3.0 `tikv_gc_mode = distributed` `tikv_gc_auto_concurrency = true` (default)	Auto-concurrent	Distributed

- Serial: requests are sent from TiDB Region by Region.
- Concurrent: requests are sent to each Region concurrently based on the number of threads specified in the `tikv_gc_concurrency`.
- Auto-concurrent: requests are sent to each Region concurrently with the number of TiKV nodes as concurrency value.
- Distributed: no need for TiDB to send requests to TiKV to trigger GC because each TiKV handles GC on its own.

4.8.2.8 GC I/O limit

TiKV supports the GC I/O limit New in v3.0.6. You can configure `gc.max-write-bytes-per-sec` to limit writes of a GC worker per second, and thus to reduce the impact on normal requests.

`0` indicates disabling this feature.

You can dynamically modify this configuration using tikv-ctl:

```
tikv-ctl --host=ip:port modify-tikv-config -m server -n gc.
    ↪ max_write_bytes_per_sec -v 10MB
```

4.9 Performance

4.9.1 SQL Optimization Process

In TiDB, the process of SQL optimization consists of two phases: logical optimization and physical optimization. This document describes the logical and physical optimization to help you understand the whole process.

4.9.1.1 Logical optimization

Based on rules, logical optimization applies some optimization rules to the input logical execution plan in order, to make the whole logical execution plan better. The optimization rules include:

- Column pruning
- Eliminate projection
- Decorrelate correlated subqueries
- Eliminate Max/Min
- Push down predicates
- Partition pruning
- Push down TopN and Limit

4.9.1.2 Physical optimization

Based on cost, physical optimization makes the physical execution plan for the logical execution plan generated in the previous phase.

In this phase, the optimizer selects the specific physical implementation for each operator in the logical execution plan. Different physical implementations of logical operators differ in time complexity, resource consumption, physical properties, and so on. During this process, the optimizer determines the cost of different physical implementations according to data statistics, and selects the physical execution plan with the minimum whole cost.

The logical execution plan is a tree structure and each node corresponds to a logical operator in SQL. Similarly, the physical execution plan is also a tree structure, and each node corresponds to a physical operator in SQL.

The logical operator only describes the function of an operator, while the physical operator describes the concrete algorithm that implements this function. A single logical operator might have multiple physical operator implementations. For example, to implement `LogicalAggregate`, you can use either `HashAggregate` the of the hash algorithm, or `StreamAggregate` of the stream type.

Different physical operators have different physical properties, and have different requirements on the physical properties of their subnodes. The physical properties include the data's order, distribution, and so on. Currently, only the data order is considered in TiDB.

4.9.2 Understand the Query Execution Plan

Based on the details of your tables, the TiDB optimizer chooses the most efficient query execution plan, which consists of a series of operators. This document details the execution plan information returned by the `EXPLAIN` statement in TiDB.

> **Note:**
>
> You are reading an earlier version of Understand the Query Execution Plan (TiDB v3.0). `EXPLAIN` Overview is the latest version of this document and is recommended if you do not have any special needs.

4.9.2.1 Optimize SQL statements using `EXPLAIN`

The result of the `EXPLAIN` statement provides information about how TiDB executes SQL queries:

- `EXPLAIN` works together with `SELECT`, `DELETE`, `INSERT`, `REPLACE`, and `UPDATE`.
- When you run the `EXPLAIN` statement, TiDB returns the final physical execution plan which is optimized by the SQL statement of `EXPLAIN`. In other words, `EXPLAIN` displays the complete information about how TiDB executes the SQL statement, such as in which order, how tables are joined, and what the expression tree looks like. For more information, see `EXPLAIN` output format.
- TiDB also supports `EXPLAIN [options] FOR CONNECTION connection_id`, but with minor differences from MySQL. See `EXPLAIN FOR CONNECTION`.

The results of `EXPLAIN` shed light on how to index the data tables so that the execution plan can use the index to speed up the execution of SQL statements. You can also use `EXPLAIN` to check if the optimizer chooses the optimal order to join tables.

4.9.2.2 `EXPLAIN` output format

Currently, the `EXPLAIN` statement returns the following four columns: id, count, task, operator info. Each operator in the execution plan is described by the four properties. In the results returned by `EXPLAIN`, each row describes an operator. See the following table for details:

Property Name	Description
id	The id of an operator, to identify the uniqueness of an operator in the entire execution plan. As of TiDB 2.1, the id includes formatting to show a tree structure of operators. The data flows from a child to its parent, and each operator has one and only one parent.
count	An estimation of the number of data items that the current operator outputs, based on the statistics and the execution logic of the operator

Property Name	Description
task	the task that the current operator belongs to. The current execution plan contains two types of tasks: 1) the **root** task that runs on the TiDB server; 2) the **cop** task that runs concurrently on the TiKV server. The topological relations of the current execution plan in the task level is that a root task can be followed by many cop tasks. The root task uses the output of cop task as the input. The cop task executes the tasks that TiDB pushes to TiKV. Each cop task scatters in the TiKV cluster and is executed by multiple processes.

Property Name	Description
operator info	The details about each operator. The information of each operator differs from others, see Operator Info.

4.9.2.2.1 Example usage

Using the bikeshare example database:

```
mysql> EXPLAIN SELECT count(*) FROM trips WHERE start_date BETWEEN '
    ↪ 2017-07-01 00:00:00' AND '2017-07-01 23:59:59';
+--
    ↪ ----------------------+-------------+------+-----------------------------
    ↪
| id                    | count      | task | operator info
    ↪
    ↪ |
+--
    ↪ ----------------------+-------------+------+-----------------------------
    ↪
| StreamAgg_20          | 1.00       | root | funcs:count(col_0)
    ↪
    ↪ |
|  ─TableReader_21      | 1.00       | root | data:StreamAgg_9
    ↪
    ↪ |
|    ─StreamAgg_9       | 1.00       | cop | funcs:count(1)
    ↪
    ↪ |
|      ─Selection_19    | 8166.73    | cop | ge(bikeshare.trips.start_date,
    ↪ 2017-07-01 00:00:00.000000), le(bikeshare.trips.start_date,
    ↪ 2017-07-01 23:59:59.000000) |
|        ─TableScan_18  | 19117643.00 | cop | table:trips, range:[-inf,+inf
    ↪ ], keep order:false
    ↪                                                              |
+--
    ↪ ----------------------+-------------+------+-----------------------------
    ↪
5 rows in set (0.00 sec)
```

Here you can see that the coprocesor (cop) needs to scan the table `trips` to find rows that match the criteria of `start_date`. Rows that meet this criteria are determined in `Selection_19` and passed to `StreamAgg_9`, all still within the coprocessor (i.e. inside of TiKV). The `count` column shows an approximate number of rows that will be processed, which is estimated with the help of table statistics. In this query it is estimated that each of the TiKV nodes will return `1.00` row to TiDB (as `TableReader_21`), which are then aggregated as `StreamAgg_20` to return an estimated `1.00` row to the client.

The good news with this query is that most of the work is pushed down to the coprocessor. This means that minimal data transfer is required for query execution. However, the `TableScan_18` can be eliminated by adding an index to speed up queries on `start_date`:

```
mysql> ALTER TABLE trips ADD INDEX (start_date);
..
mysql> EXPLAIN SELECT count(*) FROM trips WHERE start_date BETWEEN '
    ↪ 2017-07-01 00:00:00' AND '2017-07-01 23:59:59';
+--
    ↪ ---------------------+---------+------+-----------------------------------------
    ↪
| id                    | count   | task | operator info
    ↪
    ↪ |
+--
    ↪ ---------------------+---------+------+-----------------------------------------
    ↪
| StreamAgg_25          | 1.00    | root | funcs:count(col_0)
    ↪                                                                              |
|  ─IndexReader_26      | 1.00    | root | index:StreamAgg_9
    ↪                                                                               |
|    ─StreamAgg_9       | 1.00    | cop | funcs:count(1)
    ↪
    ↪ |
|      ─IndexScan_24    | 8166.73 | cop | table:trips, index:start_date,
    ↪ range:[2017-07-01 00:00:00,2017-07-01 23:59:59], keep order:false |
+--
    ↪ ---------------------+---------+------+-----------------------------------------
    ↪
4 rows in set (0.01 sec)
```

In the revisited `EXPLAIN` you can see the count of rows scanned has reduced via the use of an index. On a reference system, the query execution time reduced from 50.41 seconds to 0.01 seconds!

4.9.2.3 `EXPLAIN ANALYZE` output format

As an extension to `EXPLAIN`, `EXPLAIN ANALYZE` will execute the query and provide additional execution statistics in the `execution info` column as follows:

- `time` shows the total wall time from entering the operator until exiting the execution. It includes all execution time of any child operator operations. If the operator is called multiple times (`loops`) from a parent operator, the time will be the cumulative time.
- `loops` is the number of times the operator was called from the parent operator.
- `rows` is the total number of rows that were returned by this operator. So for example, you can compare the accuracy of the `count` column to `rows` in the `execution_info` column to assess how accurate the query optimizer's estimations are.

4.9.2.3.1 Example usage

```
mysql> EXPLAIN ANALYZE SELECT count(*) FROM trips WHERE start_date BETWEEN
    ↪ '2017-07-01 00:00:00' AND '2017-07-01 23:59:59';
+--
    ↪ ---------------------+---------+------+-----------------------------------
    ↪
| id                    | count   | task | operator info
    ↪
    ↪ | execution info                |
+--
    ↪ ---------------------+---------+------+-----------------------------------
    ↪
| StreamAgg_25          | 1.00    | root | funcs:count(col_0)
    ↪                                                                          |
    ↪ time:79.851424ms, loops:2, rows:1 |
|  ─IndexReader_26      | 1.00    | root | index:StreamAgg_9
    ↪                                                                           |
    ↪ time:79.835575ms, loops:2, rows:1 |
|    ─StreamAgg_9       | 1.00    | cop | funcs:count(1)
    ↪
    ↪ |                                 |
|      ─IndexScan_24    | 8161.83 | cop | table:trips, index:start_date,
    ↪ range:[2017-07-01 00:00:00,2017-07-01 23:59:59], keep order:false |
    ↪                   |
+--
    ↪ ---------------------+---------+------+-----------------------------------
    ↪
4 rows in set (0.08 sec)
```

4.9.2.4 `EXPLAIN FOR CONNECTION`

`EXPLAIN FOR CONNECTION` provides the last query execution plan used by a given connection. The output format is totally the same as `EXPLAIN` in TiDB. The usage has the following semantic differences from MySQL:

- MySQL returns the plan for the currently **executing query** while TiDB returns the execution plan for the **last executed** query.
- In MySQL, executing `EXPLAIN FOR CONNECTION` for connections owned by other users requires the `PROCESS` privilege. In TiDB, this statement requires the `SUPER` privilege.

4.9.2.5 Overview

4.9.2.5.1 Introduction to task

Currently, there are two types of task execution: cop tasks and root tasks. A cop task refers to a computing task that is executed using the TiKV coprocessor. A root task refers to a computing task that is executed in TiDB.

One of the goals of SQL optimization is to push the calculation down to TiKV as much as possible. The TiKV coprocessor is able to assist in execution of SQL functions (both aggregate and scalar), SQL `LIMIT` operations, index scans, and table scans. All join operations, however, will be performed as root tasks.

4.9.2.5.2 Table data and index data

The table data in TiDB refers to the raw data of a table, which is stored in TiKV. For each row of the table data, its key is a 64-bit integer called Handle ID. If a table has int type primary key, the value of the primary key is taken as the Handle ID of the table data, otherwise the system automatically generates the Handle ID. The value of the table data is encoded by all the data in this row. When the table data is read, return the results in the order in which the Handle ID is incremented.

Similar to the table data, the index data in TiDB is also stored in TiKV. The key of index data is ordered bytes encoded by index columns. The value is the Handle ID of each row of index data. You can use the Handle ID to read the non-index columns in this row. When the index data is read, return the results in the order in which the index columns are incremented. If the case of multiple index columns, make sure that the first column is incremented and that the i + 1 column is incremented when the i column is equal.

4.9.2.5.3 Range query

In the WHERE/HAVING/ON condition, analyze the results returned by primary key or index key queries. For example, number and date types of comparison symbols, greater than, less than, equal to, greater than or equal to, less than or equal to, and character type LIKE symbols.

TiDB only supports the comparison symbols of which one side is a column and the other side is a constant or can be calculated as a constant. Query conditions like `year(birth_day )< 1992` cannot use the index. Try to use the same type to compare: additional cast operations prevent the index from being used. For example, in `user_id = 123456`, if the `user_id` is a string, you need to write `123456` as a string constant.

Using `AND` and `OR` combination on the range query conditions of the same column is equivalent to getting the intersection or union set. For multidimensional combined indexes, you can write the conditions for multiple columns. For example, in the `(a, b, c)` combined index, when `a` is an equivalent query, you can continue to calculate the query range of `b` ; when `b` is also an equivalent query, you can continue to calculate the query range of `c`; otherwise, if `a` is a non-equivalent query, you can only calculate the query range of `a`.

4.9.2.6 Operator info

4.9.2.6.1 TableReader and TableScan

TableScan refers to scanning the table data at the KV side. TableReader refers to reading the table data from TiKV at the TiDB side. TableReader and TableScan are the two operators of one function. The `table` represents the table name in SQL statements. If the table is renamed, it displays the new name. The `range` represents the range of scanned data. If the WHERE/HAVING/ON condition is not specified in the query, full table scan is executed. If the range query condition is specified on the int type primary keys, range query is executed. The `keep order` indicates whether the table scan is returned in order.

4.9.2.6.2 IndexReader and IndexLookUp

The index data in TiDB is read in two ways: 1) IndexReader represents reading the index columns directly from the index, which is used when only index related columns or primary keys are quoted in SQL statements; 2) IndexLookUp represents filtering part of the data from the index, returning only the Handle ID, and retrieving the table data again using Handle ID. In the second way, data is retrieved twice from TiKV. The way of reading index data is automatically selected by the optimizer.

Similar to TableScan, IndexScan is the operator to read index data in the KV side. The `table` represents the table name in SQL statements. If the table is renamed, it displays the new name. The `index` represents the index name. The `range` represents the range of scanned data. The `out of order` indicates whether the index scan is returned in order. In TiDB, the primary key composed of multiple columns or non-int columns is treated as the unique index.

4.9.2.6.3 Selection

Selection represents the selection conditions in SQL statements, usually used in WHERE/HAVING/ON clause.

4.9.2.6.4 Projection

Projection corresponds to the `SELECT` list in SQL statements, used to map the input data into new output data.

4.9.2.6.5 Aggregation

Aggregation corresponds to `Group By` in SQL statements, or the aggregate functions if the `Group By` statement does not exist, such as the `COUNT` or `SUM` function. TiDB supports two aggregation algorithms: Hash Aggregation and Stream Aggregation. Hash Aggregation is a hash-based aggregation algorithm. If Hash Aggregation is close to the read operator of Table or Index, the aggregation operator pre-aggregates in TiKV to improve the concurrency and reduce the network load.

4.9.2.6.6 Join

TiDB supports Inner Join and Left/Right Outer Join, and automatically converts the external connection that can be simplified to Inner Join.

TiDB supports three Join algorithms: Hash Join, Sort Merge Join and Index Look up Join. The principle of Hash Join is to pre-load the memory with small tables involved in the connection and read all the data of big tables to connect. The principle of Sort Merge Join is to read the data of two tables at the same time and compare one by one using the order information of the input data. Index Look Up Join reads data of external tables and executes primary key or index key queries on internal tables.

4.9.2.6.7 Apply

Apply is an operator used to describe subqueries in TiDB. The behavior of Apply is similar to Nested Loop. The Apply operator retrieves one piece of data from external tables, puts it into the associated column of the internal tables, executes and calculates the connection according to the inline Join algorithm in Apply.

Generally, the Apply operator is automatically converted to a Join operation by the query optimizer. Therefore, try to avoid using the Apply operator when you write SQL statements.

4.9.3 The Blocklist of Optimization Rules and Expression Pushdown

This document introduces how to use the blocklist of optimization rules and the blocklist of expression pushdown to control the behavior of TiDB.

4.9.3.1 The blocklist of optimization rules

The blocklist of optimization rules is one way to tune optimization rules, mainly used to manually disable some optimization rules.

4.9.3.1.1 Important optimization rules

Optimization Rule	Rule Name	Description
Column pruning	column_prune	Prune operator will prune the column if it is not needed by the upper executor.
Decorrelate subquery	decorrelate	Tries to rewrite the correlated subquery to non-correlated join or aggregation.

Optimization Rule	Rule Name	Description
Aggregation elimination	aggregation_eliminate	Tries to remove unnecessary aggregation operators from the execution plan.
Projection elimination	projection_eliminate	Removes unnecessary projection operators from the execution plan.

Optimization Rule	Rule Name	Description
Max/Min elimination	max_min_eliminate	Revolutionize some max/min functions in aggregation to the `order by` + `limit 1` form.
Predicate push-down	predicate_push_down	Try to push predicates down to the operator that is closer to the data source.

Optimization Rule	Rule Name	Description
Outer join elimination	outer_join_elimination	Tries to eliminate the unnecessary left join or right join from the execution plan.

Optimization Rule	Rule Name	Description
Partition pruning	partition_processor	Prunes partitions which are rejected by the predicates and rewrite partitioned table query to the `UnionAll` ↪ ↪ `+` ↪ ↪ `Partition` ↪ ↪ `Datasource` ↪ form.
Aggregation push-down	aggregation_push_down	Tries to push aggregations down to their children.

Optimization Rule	Rule Name	Description
TopN push-down	topn_push_down	This rule is to push the TopN operator to the place closer to the data source.
Join reorder	join_reorder	Decides the order of multi-table joins.

4.9.3.1.2 Disable optimization rules

You can use the blocklist of optimization rules to disable some of them if some rules lead to a sub-optimal execution plan for special queries.

Usage

> **Note:**
>
> All the following operations need the `super privilege` privilege of the database. Each optimization rule has a name. For example, the name of column pruning is `column_prune`. The names of all optimization rules can be found in the second column of the table Important Optimization Rules.

- If you want to disable some rules, write its name to the `mysql.opt_rule_blacklist` table. For example:

```
INSERT INTO mysql.opt_rule_blacklist VALUES("join_reorder"), ("
    ↪ topn_push_down");
```

Executing the following SQL statement can make the above operation take effect immediately. The effective range includes all old connections of the corresponding TiDB server:

```
admin reload opt_rule_blacklist;
```

> **Note:**
>
> `admin reload opt_rule_blacklist` only takes effect on the TiDB server where the above statement has been run. If you want all TiDB servers of the cluster to take effect, run this command on each TiDB server.

- If you want to re-enable a rule, delete the corresponding data in the table, and then run the `admin reload` statement:

```
DELETE FROM mysql.opt_rule_blacklist WHERE name IN ("join_reorder", "
    ↪ topn_push_down");
```

```
admin reload opt_rule_blacklist;
```

4.9.3.2 The blocklist of expression pushdown

The blocklist of expression pushdown is one way to tune the expression pushdown, mainly used to manually disable some expressions of some specific data types.

4.9.3.2.1 Expressions which are supported to be pushed down

Expression Classification	Concrete Operations
Logical operations	AND (&&), OR (||), NOT (!)
Comparison functions and operators	<, <=, =, != (<>), >, >=, <=>, IN(), IS NULL, LIKE, IS TRUE, IS FALSE, COALESCE()
Numeric functions and operators	+, -, *, /, ABS(), CEIL(), CEILING(), FLOOR()
Control flow functions	CASE, IF(), IFNULL()

Expression Classification	Concrete Operations
JSON functions	JSON_TYPE(json_val), JSON_EXTRACT(json_doc, path[, path] ...), JSON_UNQUOTE(json_val), JSON_OBJECT(key, val[, key, val] ...), JSON_ARRAY([val[, val] ...]), JSON_MERGE(json_doc, json_doc[, json_doc] ...), JSON_SET(json_doc, path, val[, path, val] ...), JSON_INSERT(json_doc, path, val[, path, val] ...), JSON_REPLACE(json_doc, path, val[, path, val] ...), JSON_REMOVE(json_doc, path[, path] ...)
Date and time functions	DATE_FORMAT()

4.9.3.2.2 Disable the pushdown of specific expressions

When you get wrong results due to the expression pushdown, you can use the blocklist to make a quick recovery for the application. More specifically, you can add some of the supported functions or operators to the `mysql.expr_pushdown_blacklist` table to disable the pushdown of specific expressions.

The schema of `mysql.expr_pushdown_blacklist` is shown as follows:

```
DESC mysql.expr_pushdown_blacklist;
```

```
+------------+-------------+------+------+-------------------+-------+
| Field      | Type        | Null | Key | Default           | Extra |
+------------+-------------+------+------+-------------------+-------+
| name       | char(100)   | NO   |     | NULL              |       |
| store_type | char(100)   | NO   |     | tikv,tiflash,tidb |       |
| reason     | varchar(200) | YES |    | NULL              |       |
+------------+-------------+------+------+-------------------+-------+
3 rows in set (0.00 sec)
```

Here is the description of each field above:

- `name`: The name of the function that is disabled to be pushed down.
- `store_type`: To specify the component that you want to prevent the function from being pushed down to for computing. Available components are `tidb`, `tikv`, and `tiflash`. The `store_type` is case-insensitive. If you need to specify multiple components, use a comma to separate each component.

  - When `store_type` is `tidb`, it indicates whether the function can be executed in other TiDB servers while the TiDB memory table is being read.
  - When `store_type` is `tikv`, it indicates whether the function can be executed in TiKV server's Coprocessor component.
  - When `store_type` is `tiflash`, it indicates whether the function can be executed in TiFlash Server's Coprocessor component.

- `reason`: To record the reason why this function is added to the blocklist.

4.9.3.2.3 Usage

This section describes how to use the blocklist of expression pushdown.

Add to the blocklist

To add one or more expressions (functions or operators) to the blocklist, perform the following steps:

1. Insert the corresponding function name or operator name, and the set of components you want to disable the pushdown, to the `mysql.expr_pushdown_blacklist` table.
2. Execute `admin reload expr_pushdown_blacklist`.

4.9.3.2.4 Remove from the blocklist

To remove one or more expressions from the blocklist, perform the following steps:

1. Delete the corresponding function name or operator name, and the set of components you want to disable the pushdown, from the `mysql.expr_pushdown_blacklist` table.
2. Execute `admin reload expr_pushdown_blacklist`.

> **Note:**
>
> `admin reload expr_pushdown_blacklist` only takes effect on the TiDB server where this statement is run. If you want all TiDB servers of the cluster to take effect, run this command on each TiDB server.

4.9.3.3 Expression blocklist usage example

In the following example, the `<` and `>` operators are added to the blocklist, and then the `>` operator is removed from the blocklist.

To judge whether the blocklist takes effect, observe the results of `EXPLAIN` (See Optimize SQL statements using `EXPLAIN`).

1. The predicates `a < 2` and `a > 2` in the `WHERE` clause of the following SQL statement can be pushed down to TiKV.

```
EXPLAIN SELECT * FROM t WHERE a < 2 AND a > 2;
```

```
+--
    ↪ ----------------------+---------+----------+---------------+-----------
    ↪
| id                    | estRows | task     | access object | operator
    ↪ info                |
+--
    ↪ ----------------------+---------+----------+---------------+-----------
    ↪
| TableReader_7         | 0.00    | root     |               | data:
    ↪ Selection_6               |
| └─Selection_6         | 0.00    | cop[tikv] |               | gt(ssb_1.t.
    ↪ a, 2), lt(ssb_1.t.a, 2) |
|   └─TableFullScan_5   | 10000.00 | cop[tikv] | table:t   | keep order:
    ↪ false, stats:pseudo |
+--
    ↪ ----------------------+---------+----------+---------------+-----------
    ↪
3 rows in set (0.00 sec)
```

2. Insert the expression to the `mysql.expr_pushdown_blacklist` table and execute `admin reload expr_pushdown_blacklist`.

```
INSERT INTO mysql.expr_pushdown_blacklist VALUES('<','tikv',''), ('>',
    ↪ 'tikv','');
```

```
Query OK, 2 rows affected (0.01 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

```
admin reload expr_pushdown_blacklist;
```

```
Query OK, 0 rows affected (0.00 sec)
```

3. Observe the execution plan again and you will find that both the `<` and `>` operators are not pushed down to TiKV Coprocessor.

```
EXPLAIN SELECT * FROM t WHERE a < 2 and a > 2;
```

```
+--
    ↪ ----------------------+---------+----------+---------------+-----------
    ↪
```

```
| id                    | estRows | task     | access object | operator
    ↪ info                |
+--
    ↪ ----------------------+---------+----------+---------------+----------
    ↪
| Selection_7           | 10000.00 | root    |               | gt(ssb_1.t.a
    ↪ , 2), lt(ssb_1.t.a, 2) |
|  ─TableReader_6       | 10000.00 | root    |               | data:
    ↪ TableFullScan_5           |
|    ─TableFullScan_5   | 10000.00 | cop[tikv] | table:t   | keep order:
    ↪ false, stats:pseudo |
+--
    ↪ ----------------------+---------+----------+---------------+----------
    ↪
3 rows in set (0.00 sec)
```

4. Remove one expression (here is >) from the blocklist and execute `admin reload ↪ expr_pushdown_blacklist`.

```
DELETE FROM mysql.expr_pushdown_blacklist WHERE name = '>';
```

```
Query OK, 1 row affected (0.01 sec)
```

```
admin reload expr_pushdown_blacklist;
```

```
Query OK, 0 rows affected (0.00 sec)
```

5. Observe the execution plan again and you will find that < is not pushed down while > is pushed down to TiKV Coprocessor.

```
EXPLAIN SELECT * FROM t WHERE a < 2 AND a > 2;
```

```
+--
    ↪ ----------------------+---------+----------+---------------+----------
    ↪
| id                     | estRows | task     | access object | operator
    ↪  info             |
+--
    ↪ ----------------------+---------+----------+---------------+----------
    ↪
| Selection_8            | 0.00    | root     |               | lt(ssb_1.t
    ↪ .a, 2)            |
|  ─TableReader_7        | 0.00    | root     |               | data:
    ↪ Selection_6           |
```

```
|   └─Selection_6         | 0.00    | cop[tikv] |           | gt(ssb_1.
    ↪ t.a, 2)           |
|     └─TableFullScan_5   | 10000.00 | cop[tikv] | table:t   | keep
    ↪ order:false, stats:pseudo |
+--
    ↪ ------------------------+----------+-----------+---------------+----------
    ↪
4 rows in set (0.00 sec)
```

4.9.4 Introduction to Statistics

Based on the statistics, the TiDB optimizer chooses the most efficient query execution plan. The statistics collect table-level and column-level information.

- The statistics of a table include the total number of rows and the number of updated rows.
- The statistics of a column include the number of different values, the number of `NULL`, the histogram, the `TOPN` value that occurs most frequently in the column, and the Count-Min Sketch of the column.

4.9.4.1 Collect statistics

4.9.4.1.1 Manual collection

You can run the `ANALYZE` statement to collect statistics.

> **Note:**
>
> The execution time of `ANALYZE TABLE` in TiDB is longer than that in MySQL or InnoDB. In InnoDB, only a small number of pages are sampled, while in TiDB a comprehensive set of statistics is completely rebuilt. Scripts that were written for MySQL may naively expect `ANALYZE TABLE` will be a short-lived operation.
>
> For quicker analysis, you can set `tidb_enable_fast_analyze` to `1` to enable the Quick Analysis feature. The default value for this parameter is `0`.
>
> After Quick Analysis is enabled, TiDB randomly samples approximately 10,000 rows of data to build statistics. Therefore, in the case of uneven data distribution or a relatively small amount of data, the accuracy of statistical information is relatively poor. It might lead to poor execution plans, such as choosing the wrong index. If the execution time of the normal `ANALYZE` ↪ statement is acceptable, it is recommended to disable the Quick Analysis feature.

Full collection

You can perform full collection using the following syntax.

- To collect statistics of all the tables in `TableNameList`:

```
ANALYZE TABLE TableNameList [WITH NUM BUCKETS];
```

`WITH NUM BUCKETS` specifies the maximum number of buckets in the generated histogram.

- To collect statistics of the index columns on all `IndexNameLists` in `TableName`:

```
ANALYZE TABLE TableName INDEX [IndexNameList] [WITH NUM BUCKETS];
```

The statement collects statistics of all index columns when `IndexNameList` is empty.

- To collect statistics of partition in all `PartitionNameLists` in `TableName`:

```
ANALYZE TABLE TableName PARTITION PartitionNameList [WITH NUM BUCKETS];
```

- To collect statistics of index columns for the partitions in all `PartitionNameLists` in `TableName`:

```
ANALYZE TABLE TableName PARTITION PartitionNameList INDEX [
    ↪ IndexNameList] [WITH NUM BUCKETS];
```

Incremental collection

To improve the speed of analysis after full collection, incremental collection could be used to analyze the newly added sections in monotonically non-decreasing columns such as time columns.

Note:

- Currently, the incremental collection is only provided for index.
- When using the incremental collection, you must ensure that only `INSERT` operations exist on the table, and that the newly inserted value on the index column is monotonically non-decreasing. Otherwise, the statistical information might be inaccurate, affecting the TiDB optimizer to select an appropriate execution plan.

You can perform incremental collection using the following syntax.

- To incrementally collect statistics for index columns in all `IndexNameLists` in `TableName`:

```
ANALYZE INCREMENTAL TABLE TableName INDEX [IndexNameList] [WITH NUM
    ↪ BUCKETS];
```

- To incrementally collect statistics of index columns for partitions in all `PartitionNameLists` ↪ in `TableName`:

```
ANALYZE INCREMENTAL TABLE TableName PARTITION PartitionNameList INDEX [
    ↪ IndexNameList] [WITH NUM BUCKETS];
```

4.9.4.1.2 Automatic update

For the `INSERT`, `DELETE`, or `UPDATE` statements, TiDB automatically updates the number of rows and updated rows. TiDB persists this information regularly and the update cycle is 20 * `stats-lease`. The default value of `stats-lease` is `3s`. If you specify the value as `0`, it does not update automatically.

Three system variables related to automatic update of statistics are as follows:

System Variable	Default Value	Description
`tidb_auto_analyze_ratio` ↪	0.5	The threshold value of automatic update

System Variable	Default Value	Description
tidb_auto_analyze_start_time	00:00 +0000	The start time in a day when TiDB can perform automatic update
tidb_auto_analyze_end_time	23:59 +0000	The end time in a day when TiDB can perform automatic update

When the ratio of the number of modified rows to the total number of rows of `tbl` in a table is greater than `tidb_auto_analyze_ratio`, and the current time is between `tidb_auto_analyze_start_time` and `tidb_auto_analyze_end_time`, TiDB executes the `ANALYZE TABLE tbl` statement in the background to automatically update the statistics of this table.

When the query is executed, TiDB collects feedback with the probability of `feedback-probability` and uses it to update the histogram and Count-Min Sketch. You can modify the value of `feedback-probability` in the configuration file. The default value is `0.05`. You can set the value to `0.0` to disable this feature.

Note:

If you set the value of `feedback-probability` to `0` in the configuration file, a failure will occur and an error will be reported. To disable `feedback-probability`, you need to set the value to `0.0`.

4.9.4.1.3 Control `ANALYZE` concurrency

When you run the `ANALYZE` statement, you can adjust the concurrency using the following parameters, to control its effect on the system.

`tidb_build_stats_concurrency`

Currently, when you run the `ANALYZE` statement, the task is divided into multiple small tasks. Each task only works on one column or index. You can use the `tidb_build_stats_concurrency` parameter to control the number of simultaneous tasks. The default value is `4`.

`tidb_distsql_scan_concurrency`

When you analyze regular columns, you can use the `tidb_distsql_scan_concurrency` parameter to control the number of Region to be read at one time. The default value is `15`.

`tidb_index_serial_scan_concurrency`

When you analyze index columns, you can use the `tidb_index_serial_scan_concurrency` parameter to control the number of Region to be read at one time. The default value is `1`.

4.9.4.1.4 View `ANALYZE` state

When executing the `ANALYZE` statement, you can view the current state of `ANALYZE` using the following SQL statement:

```
SHOW ANALYZE STATUS [ShowLikeOrWhere]
```

This statement returns the state of `ANALYZE`. You can use `ShowLikeOrWhere` to filter the information you need.

Currently, the `SHOW ANALYZE STATUS` statement returns the following 7 columns:

Syntax Element	Description
table_schema	The database name
table_name	The table name

Syntax Element	Description
partition_name	The partition name
job_info	The task information. The element includes index names when index analysis is performed.
row_count	The number of rows that have been analyzed
start_time	The time at which the task starts
state	The state of a task, including `pending`, `running`, `finished`, and `failed`

4.9.4.2 View statistics

You can view the statistics status using the following statements.

4.9.4.2.1 Metadata of tables

You can use the `SHOW STATS_META` statement to view the total number of rows and the number of updated rows.

Syntax as follows:

```
SHOW STATS_META [ShowLikeOrWhere]
```

This statement returns the total number of all the rows in all the tables and the number of updated rows. You can use `ShowLikeOrWhere` to filter the information you need.

Currently, the `SHOW STATS_META` statement returns the following 6 columns:

Syntax Element	Description
`db_name`	The database name
`table_name`	The table name
`partition_name`	The partition name

Syntax Element	Description
`update_time`	The the time of the update
`modify_count`	The number of modified rows
`row_count`	The total number of rows

Note:

When TiDB automatically updates the total number of rows and the number of modified rows according to DML statements, `update_time` is also updated. Therefore, `update_time` does not necessarily indicate the last time when the `ANALYZE` statement is executed.

4.9.4.2.2 Health state of tables

You can use the `SHOW STATS_HEALTHY` statement to check the health state of tables and roughly estimate the accuracy of the statistics. When `modify_count >= row_count`, the health state is 0; when `modify_count < row_count`, the health state is (1 - `modify_count` ↪ `/row_count`) * 100.

The syntax is as follows. You can use `ShowLikeOrWhere` to filter the information you need:

```
SHOW STATS_HEALTHY [ShowLikeOrWhere];
```

Currently, the `SHOW STATS_HEALTHY` statement returns the following 4 columns:

Syntax Element	Description
`db_name`	The database name
`table_name`	The table name
`partition_name`	The partition name
`healthy`	The health state of tables

4.9.4.2.3 Metadata of columns

You can use the `SHOW STATS_HISTOGRAMS` statement to view the number of different values and the number of `NULL` in all the columns.

Syntax as follows:

```
SHOW STATS_HISTOGRAMS [ShowLikeOrWhere]
```

This statement returns the number of different values and the number of `NULL` in all the columns. You can use `ShowLikeOrWhere` to filter the information you need.

Currently, the `SHOW STATS_HISTOGRAMS` statement returns the following 8 columns:

Syntax Element	Description
`db_name`	The database name
`table_name`	The table name
`partition_name`	The partition name
`column_name`	The column name (when `is_index` is `0`) or the index name (when `is_index` is `1`)
`is_index`	Whether it is an index column or not
`update_time`	The time of the update
`distinct_count`	The number of different values
`null_count`	The number of `NULL`
`avg_col_size`	The average length of columns

4.9.4.2.4 Buckets of histogram

You can use the `SHOW STATS_BUCKETS` statement to view each bucket of the histogram.

Syntax as follows:

```
SHOW STATS_BUCKETS [ShowLikeOrWhere]
```

This statement returns information about all the buckets. You can use `ShowLikeOrWhere` to filter the information you need.

Currently, the `SHOW STATS_BUCKETS` statement returns the following 10 columns:

Syntax Element	Description
`db_name`	The database name

Syntax Element	Description
`table_name`	The table name
`partition_name`	The partition name
`column_name`	The column name (when `is_index` is `0`) or the index name (when `is_index` is `1`)
`is_index`	Whether it is an index column or not
`bucket_id`	The the ID of a bucket
`count`	The number of all the values that falls on the bucket and the previous buckets
`repeats`	The occurrence number of the maximum value
`lower_bound`	The minimum value
`upper_bound`	The maximum value

4.9.4.3 Delete statistics

You can run the `DROP STATS` statement to delete statistics.

Syntax as follows:

```
DROP STATS TableName
```

The statement deletes statistics of all the tables in `TableName`.

4.9.4.4 Import and export statistics

4.9.4.4.1 Export statistics

The interface to export statistics is as follows:

- To obtain the JSON format statistics of the `${table_name}` table in the `${db_name}` database:

```
http://${tidb-server-ip}:${tidb-server-status-port}/stats/dump/${
    ↪ db_name}/${table_name}
```

- To obtain the JSON format statistics of the `${table_name}` table in the `${db_name}` database at specific time:

```
http://${tidb-server-ip}:${tidb-server-status-port}/stats/dump/${
    ↪ db_name}/${table_name}/${yyyyMMddHHmmss}
```

4.9.4.4.2 Import statistics

Generally, the imported statistics refer to the JSON file obtained using the export interface.

Syntax:

```
LOAD STATS 'file_name'
```

`file_name` is the file name of the statistics to be imported.

4.9.5 TopN and Limit Operator Push Down

This document describes the implementation of TopN and Limit operator pushdown.

In the TiDB execution plan tree, the `LIMIT` clause in SQL corresponds to the Limit operator node, and the `ORDER BY` clause corresponds to the Sort operator node. The adjacent Limit operator and Sort operator are combined as the TopN operator node, which means that the top N records are returned according to a certain sorting rule. That is to say, a Limit operator is equivalent to a TopN operator node with a null sorting rule.

Similar to predicate pushdown, TopN and Limit are pushed down in the execution plan tree to a position as close to the data source as possible so that the required data is filtered at an early stage. In this way, the pushdown significantly reduces the overhead of data transmission and calculation.

To disable this rule, refer to Optimization Rules and Blocklist for Expression Pushdown.

4.9.5.1 Examples

This section illustrates TopN pushdown through some examples.

4.9.5.1.1 Example 1: Push down to the Coprocessors in the storage layer

```
create table t(id int primary key, a int not null);
explain select * from t order by a limit 10;
```

```
+---------------------------+---------+----------+--------------+-------------------
   ↪
| id                        | estRows | task     | access object | operator
   ↪ info            |
+---------------------------+---------+----------+--------------+-------------------
   ↪
| TopN_7                    | 10.00   | root     |              | test.t.a,
   ↪ offset:0, count:10 |
|  ─TableReader_15          | 10.00   | root     |              | data:TopN_14
   ↪                |
|    ─TopN_14               | 10.00   | cop[tikv] |             | test.t.a,
   ↪ offset:0, count:10 |
|      ─TableFullScan_13    | 10000.00 | cop[tikv] | table:t   | keep order:
   ↪ false, stats:pseudo |
+---------------------------+---------+----------+--------------+-------------------
   ↪
4 rows in set (0.00 sec)
```

In this query, the TopN operator node is pushed down to TiKV for data filtering, and each Coprocessor returns only 10 records to TiDB. After TiDB aggregates the data, the final filtering is performed.

4.9.5.1.2 Example 2: TopN can be pushed down into Join (the sorting rule only depends on the columns in the outer table)

```
create table t(id int primary key, a int not null);
create table s(id int primary key, a int not null);
explain select * from t left join s on t.a = s.a order by t.a limit 10;
```

```
+---------------------------------+---------+----------+--------------+------------
   ↪
| id                              | estRows | task     | access object |
   ↪ operator info                              |
+---------------------------------+---------+----------+--------------+------------
   ↪
| TopN_12                         | 10.00   | root     |              | test.t.a,
   ↪  offset:0, count:10                |
|  ─HashJoin_17                   | 12.50   | root     |              | left
   ↪ outer join, equal:[eq(test.t.a, test.s.a)] |
|    ─TopN_18(Build)              | 10.00   | root     |              | test.t.a
   ↪ , offset:0, count:10                |
|      ─TableReader_26            | 10.00   | root     |              | data:
   ↪ TopN_25                               |
|        ─TopN_25                 | 10.00   | cop[tikv] |             | test.t.a
   ↪ , offset:0, count:10                |
```

```
|         └─TableFullScan_24     | 10000.00 | cop[tikv] | table:t   | keep
   ↪ order:false, stats:pseudo            |
|   └─TableReader_30(Probe)     | 10000.00 | root    |             | data:
   ↪ TableFullScan_29                     |
|     └─TableFullScan_29        | 10000.00 | cop[tikv] | table:s   | keep
   ↪ order:false, stats:pseudo            |
+--------------------------------+----------+-----------+--------------+------------
   ↪
8 rows in set (0.01 sec)
```

In this query, the sorting rule of the TopN operator only depends on the columns in the outer table `t`, so a calculation can be performed before pushing down TopN to Join, to reduce the calculation cost of the Join operation. Besides, TiDB also pushes TopN down to the storage layer.

4.9.5.1.3 Example 3: TopN cannot be pushed down before Join

```
create table t(id int primary key, a int not null);
create table s(id int primary key, a int not null);
explain select * from t join s on t.a = s.a order by t.id limit 10;
```

```
+------------------------------+---------+----------+--------------+---------------
   ↪
| id                          | estRows | task     | access object | operator
   ↪ info                        |
+------------------------------+---------+----------+--------------+---------------
   ↪
| TopN_12                     | 10.00   | root     |              | test.t.id,
   ↪ offset:0, count:10          |
|  └─HashJoin_16               | 12500.00 | root    |              | inner join,
   ↪  equal:[eq(test.t.a, test.s.a)] |
|    ├─TableReader_21(Build)  | 10000.00 | root    |              | data:
   ↪ TableFullScan_20                  |
|    │ └─TableFullScan_20     | 10000.00 | cop[tikv] | table:s   | keep order:
   ↪ false, stats:pseudo         |
|    └─TableReader_19(Probe)  | 10000.00 | root    |              | data:
   ↪ TableFullScan_18                  |
|      └─TableFullScan_18     | 10000.00 | cop[tikv] | table:t   | keep order:
   ↪ false, stats:pseudo         |
+------------------------------+---------+----------+--------------+---------------
   ↪
6 rows in set (0.00 sec)
```

TopN cannot be pushed down before `Inner Join`. Taking the query above as an example, if you get 100 records after Join, then you can have 10 records left after TopN. However, if

TopN is performed first to get 10 records, only 5 records are left after Join. In such cases, the pushdown results in different results.

Similarly, TopN can neither be pushed down to the inner table of Outer Join, nor can it be pushed down when its sorting rule is related to columns on multiple tables, such as `t.a+s.a`. Only when the sorting rule of TopN exclusively depends on columns on the outer table, can TopN be pushed down.

4.9.5.1.4 Example 4: Convert TopN to Limit

```
create table t(id int primary key, a int not null);
create table s(id int primary key, a int not null);
explain select * from t left join s on t.a = s.a order by t.id limit 10;
```

```
+---------------------------------+----------+-----------+---------------+------------
    ↪
| id                              | estRows | task      | access object |
    ↪ operator info                                |
+---------------------------------+----------+-----------+---------------+------------
    ↪
| TopN_12                         | 10.00   | root      |               | test.t.id
    ↪ , offset:0, count:10                 |
|  ─HashJoin_17                   | 12.50   | root      |               | left
    ↪ outer join, equal:[eq(test.t.a, test.s.a)] |
|    ─Limit_21(Build)             | 10.00   | root      |               | offset
    ↪ :0, count:10                          |
|      ─TableReader_31            | 10.00   | root      |               | data:
    ↪ Limit_30                              |
|        ─Limit_30                | 10.00   | cop[tikv] |               | offset
    ↪ :0, count:10                          |
|          ─TableFullScan_29      | 10.00   | cop[tikv] | table:t       | keep
    ↪ order:true, stats:pseudo              |
|    ─TableReader_35(Probe)       | 10000.00 | root     |               | data:
    ↪ TableFullScan_34                      |
|      ─TableFullScan_34          | 10000.00 | cop[tikv] | table:s       | keep
    ↪ order:false, stats:pseudo             |
+---------------------------------+----------+-----------+---------------+------------
    ↪
8 rows in set (0.00 sec)
```

In the query above, TopN is first pushed to the outer table `t`. TopN needs to sort by `t ↪ .id`, which is the primary key and can be directly read in order (`keep order: true`) without extra sorting in TopN. Therefore, TopN is simplified as Limit.

4.9.6 Optimizer Hints

TiDB supports optimizer hints, based on the comment-like syntax introduced in MySQL 5.7. i.e. `/*+ TIDB_XX(t1, t2)*/`. Use of optimizer hints is recommended in cases where the TiDB optimizer selects a less optimal query plan.

> **Note:**
>
> MySQL command-line clients earlier than 5.7.7 strip optimizer hints by default. If you want to use the `Hint` syntax in these earlier versions, add the `--comments` option when starting the client. For example: `mysql -h 127.0.0.1 -P 4000 -uroot --comments`.

4.9.6.1 TIDB_SMJ(t1, t2)

```
SELECT /*+ TIDB_SMJ(t1, t2) */ * from t1, t2 where t1.id = t2.id
```

This variable is used to remind the optimizer to use the `Sort Merge Join` algorithm. This algorithm takes up less memory, but takes longer to execute. It is recommended if the data size is too large, or there's insufficient system memory.

4.9.6.2 TIDB_INLJ(t1, t2)

```
SELECT /*+ TIDB_INLJ(t1, t2) */ * from t1, t2 where t1.id = t2.id
```

This variable is used to remind the optimizer to use the `Index Nested Loop Join` algorithm. In some scenarios, this algorithm runs faster and takes up fewer system resources, but may be slower and takes up more system resources in some other scenarios. You can try to use this algorithm in scenarios where the result-set is less than 10,000 rows after the outer table is filtered by the WHERE condition. The parameter in `TIDB_INLJ()` is the candidate table for the inner table when you create the query plan. For example, `TIDB_INLJ (t1)` means that TiDB only considers using t1 as the inner table to create a query plan.

4.9.6.3 TIDB_HJ(t1, t2)

```
SELECT /*+ TIDB_HJ(t1, t2) */ * from t1, t2 where t1.id = t2.id
```

This variable is used to remind the optimizer to use the `Hash Join` algorithm. This algorithm executes threads concurrently. It runs faster but takes up more memory.

4.9.6.4 MAX_EXECUTION_TIME(N)

The `MAX_EXECUTION_TIME` hint only applies to `SELECT` statements. It places a limit `N` (a timeout value in milliseconds) on how long a statement is permitted to execute before the server terminates it:

```
MAX_EXECUTION_TIME(N)
```

Example with a timeout of 1 second (1000 milliseconds):

```
SELECT /*+ MAX_EXECUTION_TIME(1000) */ * FROM t1 INNER JOIN t2 WHERE ...
```

In addition to this hint, the `max_execution_time` system variable also limits the execution time of a statement.

4.9.7 Check the TiDB Cluster Status Using SQL Statements

TiDB offers some SQL statements and system tables to check the TiDB cluster status.

The `INFORMATION_SCHEMA` system database offers system tables as follows to query the cluster status and diagnose common cluster issues:

- TABLES
- TIDB_INDEXES
- ANALYZE_STATUS
- TIDB_HOT_REGIONS
- TIKV_STORE_STATUS
- TIKV_REGION_STATUS
- TIKV_REGION_PEERS

You can also use the following statements to obtain some useful information for troubleshooting and querying the TiDB cluster status.

- `ADMIN SHOW DDL`: obtains the ID of TiDB with the `DDL owner` role and `IP:PORT`.
- The feature of `SHOW ANALYZE STATUS` is the same with that of the `ANALYZE_STATUS` table.
- Specific `EXPLAIN` statements
  - `EXPLAIN ANALYZE`: obtains some detailed information for execution of a SQL statement.
  - `EXPLAIN FOR CONNECTION`: obtains the execution plan for the query executed last in a connection. Can be used along with `SHOW PROCESSLIST`.
  - For more information about `EXPLAIN`, see Understand the Query Execution Plan.

4.9.8 Execution Plan Binding

The Optimizer Hints document introduces how to select a specific execution plan using Hint. However, sometimes you need to interfere with execution selection without modifying SQL statements. Execution Plan Binding provides a set of functionalities to do this.

4.9.8.1 Syntax

4.9.8.1.1 Create binding

```
CREATE [GLOBAL | SESSION] BINDING FOR SelectStmt USING SelectStmt
```

This statement binds SQL execution plans at the GLOBAL or SESSION level. The default scope is SESSION. The bound SQL statement is parameterized and stored in the system table. When a SQL query is processed, the corresponding optimizer hint is available as long as the parameterized SQL statement and a bound one in the system table are consistent.

`Parameterization` is a process that converts a constant in SQL to a variable parameter, with standardized processing on the spaces and line breaks in the SQL statement, for example,

```
select * from t where a > 1
-- parameterized:
select * from t where a > ?
```

Note:

The text must be the same before and after parameterization and hint removal for both the original SQL statement and the bound statement, or the binding will fail. Take the following examples:

- This binding can be created successfully because the texts before and after parameterization and hint removal are the same: `select * from t where a > ?`

```
CREATE BINDING FOR SELECT * FROM t WHERE a > 1 USING
    ↪ SELECT * FROM t use index (idx) WHERE a > 2
```

- This binding will fail because the original SQL statement is processed as `select * from t where a > ?`, while the bound SQL statement is processed differently as `select * from t where b > ?`.

```
CREATE BINDING FOR SELECT * FROM t WHERE a > 1 USING
    ↪ SELECT * FROM t use index(idx) WHERE b > 2
```

4.9.8.1.2 Remove binding

```
DROP [GLOBAL | SESSION] BINDING FOR SelectStmt
```

This statement removes a specified execution plan binding at the GLOBAL or SESSION level. The default scope is SESSION.

4.9.8.1.3 View binding

```
SHOW [GLOBAL | SESSION] BINDINGS [ShowLikeOrWhere]
```

This statement outputs the execution plan bindings at the GLOBAL or SESSION level. The default scope is SESSION. Currently `SHOW BINDINGS` outputs eight columns, as shown below:

Column Name	Note
original_sql	Original SQL statement after parameterization
bind_sql	Bound SQL statement with hints
default_db	Default database
status	Status including Using, Deleted, and Invalid
create_time	Creating time
update_time	Updating time
charset	Character set
collation	Ordering rule

4.9.9 Statement Summary Tables

To better handle SQL performance related issues, MySQL has provided statement summary tables in `performance_schema` to monitor SQL with statistics. Among these tables, `events_statements_summary_by_digest` is very useful in locating SQL problems with its abundant fields such as latency, execution times, rows scanned, and full table scans.

Starting from v3.0.4, TiDB provides the support for the `events_statements_summary_by_digest` ↪ table. Starting from v3.0.8, TiDB provides the support for the `events_statements_summary_by_diges` ↪ table. In this document, you will learn about the two tables, and how to troubleshoot SQL performance issues using these tables.

4.9.9.1 `events_statements_summary_by_digest`

`events_statements_summary_by_digest` is a system table in `performance_schema`. It groups the SQL statements by the SQL digest and the plan digest, and provides statistics for each SQL category.

The "SQL digest" here means the same as used in slow logs, which is a unique identifier calculated through normalized SQL statements. The normalization process ignores constant,

blank characters, and is case insensitive. Therefore, statements with consistent syntaxes have the same digest. For example:

```
SELECT * FROM employee WHERE id IN (1, 2, 3) AND salary BETWEEN 1000 AND
    ↪ 2000;
select * from EMPLOYEE where ID in (4, 5) and SALARY between 3000 and 4000;
```

After normalization, they are both of the following category:

```
select * from employee where id in (...) and salary between ? and ?;
```

The "plan digest" here refers to the unique identifier calculated through normalized execution plan. The normalization process ignores constants. The same SQL statements might be grouped into different categories because the same statements might have different execution plans. SQL statements of the same category have the same execution plan.

`events_statements_summary_by_digest` stores the aggregated results of SQL monitoring metrics. In general, each of the monitoring metrics includes the maximum value and average value. For example, the execution latency metric corresponds to two fields: `AVG_LATENCY` (average latency) and `MAX_LATENCY` (maximum latency).

To make sure that the monitoring metrics are up to date, data in the `events_statements_summary_by_` ↪ table is periodically cleared, and only recent aggregated results are retained and displayed. The periodical data clearing is controlled by the `tidb_stmt_summary_refresh_interval` ↪ system variable. If you happen to make a query right after the clearing, the data displayed might be very little.

Some concepts in TiDB are different from those in MySQL. For this reason, the schema of the `events_statements_summary_by_digest` table in TiDB greatly differs from that in MySQL.

The following is a sample output of querying `events_statements_summary_by_digest`:

```
  SUMMARY_BEGIN_TIME: 2020-01-02 11:00:00
    SUMMARY_END_TIME: 2020-01-02 11:30:00
           STMT_TYPE: select
         SCHEMA_NAME: test
              DIGEST: 0611
                  ↪ cc2fe792f8c146cc97d39b31d9562014cf15f8d41f23a4938ca341f54182
                  ↪
         DIGEST_TEXT: select * from employee where id = ?
         TABLE_NAMES: test.employee
         INDEX_NAMES: NULL
         SAMPLE_USER: root
          EXEC_COUNT: 3
         SUM_LATENCY: 1035161
         MAX_LATENCY: 399594
         MIN_LATENCY: 301353
         AVG_LATENCY: 345053
```

```
  AVG_PARSE_LATENCY: 57000
  MAX_PARSE_LATENCY: 57000
AVG_COMPILE_LATENCY: 175458
MAX_COMPILE_LATENCY: 175458
...........
           AVG_MEM: 103
           MAX_MEM: 103
  AVG_AFFECTED_ROWS: 0
        FIRST_SEEN: 2020-01-02 11:12:54
         LAST_SEEN: 2020-01-02 11:25:24
  QUERY_SAMPLE_TEXT: select * from employee where id=3100
   PREV_SAMPLE_TEXT:
        PLAN_DIGEST:
           ↪ f415b8d52640b535b9b12a9c148a8630d2c6d59e419aad29397842e32e8e5de3
           ↪
              PLAN: Point_Get_1   root   1      table:employee, handle:3100
```

Note:

In TiDB, the time unit of fields in statement summary tables is nanosecond (ns), whereas in MySQL the time unit is picosecond (ps).

4.9.9.1.1 Fields description

Basic fields:

- `STMT_TYPE`: SQL statement type.
- `SCHEMA_NAME`: The current schema in which SQL statements of this category are executed.
- `DIGEST`: The digest of SQL statements of this category.
- `DIGEST_TEXT`: The normalized SQL statement.
- `QUERY_SAMPLE_TEXT`: The original SQL statements of the SQL category. Only one original statement is taken.
- `TABLE_NAMES`: All tables involved in SQL statements. If there is more than one table, each is separated by a comma.
- `INDEX_NAMES`: All SQL indexes used in SQL statements. If there is more than one index, each is separated by a comma.
- `SAMPLE_USER`: The users who execute SQL statements of this category. Only one user is taken.
- `PLAN_DIGEST`: The digest of the execution plan.
- `PLAN`: The original execution plan. If there are multiple statements, the plan of only one statement is taken.

Fields related to execution time:

- `SUMMARY_BEGIN_TIME`: The beginning time of the current summary period.
- `SUMMARY_END_TIME`: The ending time of the current summary period.
- `FIRST_SEEN`: The time when SQL statements of this category are seen for the first time.
- `LAST_SEEN`: The time when SQL statements of this category are seen for the last time.

Fields related to TiDB server:

- `EXEC_COUNT`: Total execution times of SQL statements of this category.
- `SUM_LATENCY`: The total execution latency of SQL statements of this category.
- `MAX_LATENCY`: The maximum execution latency of SQL statements of this category.
- `MIN_LATENCY`: The minimum execution latency of SQL statements of this category.
- `AVG_LATENCY`: The average execution latency of SQL statements of this category.
- `AVG_PARSE_LATENCY`: The average latency of the parser.
- `MAX_PARSE_LATENCY`: The maximum latency of the parser.
- `AVG_COMPILE_LATENCY`: The average latency of the compiler.
- `MAX_COMPILE_LATENCY`: The maximum latency of the compiler.
- `AVG_MEM`: The average memory (byte) used.
- `MAX_MEM`: The maximum memory (byte) used.

Fields related to TiKV Coprocessor task:

- `COP_TASK_NUM`: The number of Coprocessor requests that a SQL statement sends.
- `AVG_COP_PROCESS_TIME`: The average execution time of Coprocessor tasks.
- `MAX_COP_PROCESS_TIME`: The maximum execution time of Coprocessor tasks.
- `MAX_COP_PROCESS_ADDRESS`: The address of the Coprocessor task with the maximum execution time.
- `AVG_COP_WAIT_TIME`: The average waiting time of Coprocessor tasks.
- `MAX_COP_WAIT_TIME`: The maximum waiting time of Coprocessor tasks.
- `MAX_COP_WAIT_ADDRESS`: The address of the Coprocessor task with the maximum waiting time.
- `AVG_PROCESS_TIME`: The average processing time of SQL statements in TiKV.
- `MAX_PROCESS_TIME`: The maximum processing time of SQL statements in TiKV.
- `AVG_WAIT_TIME`: The average waiting time of SQL statements in TiKV.
- `MAX_WAIT_TIME`: The maximum waiting time of SQL statements in TiKV.
- `AVG_BACKOFF_TIME`: The average waiting time before retry when a SQL statement encounters an error that requires a retry.
- `MAX_BACKOFF_TIME`: The maximum waiting time before retry when a SQL statement encounters an error that requires a retry.
- `AVG_TOTAL_KEYS`: The average number of keys that Coprocessor has scanned.
- `MAX_TOTAL_KEYS`: The maximum number of keys that Coprocessor has scanned.

- `AVG_PROCESSED_KEYS`: The average number of keys that Coprocessor has processed. Compared with `avg_total_keys`, `avg_processed_keys` does not include the old versions of MVCC. A great difference between `avg_total_keys` and `avg_processed_keys` indicates that many old versions exist.
- `MAX_PROCESSED_KEYS`: The maximum number of keys that Coprocessor has processed.

Transaction-related fields:

- `AVG_PREWRITE_TIME`: The average time of the prewrite phase.
- `MAX_PREWRITE_TIME`: The longest time of the prewrite phase.
- `AVG_COMMIT_TIME`: The average time of the commit phase.
- `MAX_COMMIT_TIME`: The longest time of the commit phase.
- `AVG_GET_COMMIT_TS_TIME`: The average time of getting `commit_ts`.
- `MAX_GET_COMMIT_TS_TIME`: The longest time of getting `commit_ts`.
- `AVG_COMMIT_BACKOFF_TIME`: The average waiting time before retry when a SQL statement encounters an error that requires a retry during the commit phase.
- `MAX_COMMIT_BACKOFF_TIME`: The maximum waiting time before retry when a SQL statement encounters an error that requires a retry during the commit phase.
- `AVG_RESOLVE_LOCK_TIME`: The average time for resolving lock conflicts occurred between transactions.
- `MAX_RESOLVE_LOCK_TIME`: The longest time for resolving lock conflicts occurred between transactions.
- `AVG_LOCAL_LATCH_WAIT_TIME`: The average waiting time of the local transaction.
- `MAX_LOCAL_LATCH_WAIT_TIME`: The maximum waiting time of the local transaction.
- `AVG_WRITE_KEYS`: The average count of written keys.
- `MAX_WRITE_KEYS`: The maximum count of written keys.
- `AVG_WRITE_SIZE`: The average amount of written data (in byte).
- `MAX_WRITE_SIZE`: The maximum amount of written data (in byte).
- `AVG_PREWRITE_REGIONS`: The average number of Regions involved in the prewrite phase.
- `MAX_PREWRITE_REGIONS`: The maximum number of Regions during the prewrite phase.
- `AVG_TXN_RETRY`: The average number of transaction retries.
- `MAX_TXN_RETRY`: The maximum number of transaction retries.
- `SUM_BACKOFF_TIMES`: The sum of retries when SQL statements of this category encounter errors that require a retry.
- `BACKOFF_TYPES`: All types of errors that require retries and the number of retries for each type. The format of the field is `type:number`. If there is more than one error type, each is separated by a comma, like `txnLock:2,pdRPC:1`.
- `AVG_AFFECTED_ROWS`: The average number of rows affected.
- `PREV_SAMPLE_TEXT`: When the current SQL statement is `COMMIT`, `PREV_SAMPLE_TEXT` is the previous statement to `COMMIT`. In this case, SQL statements are grouped by the digest and `prev_sample_text`. This means that `COMMIT` statements with different `prev_sample_text` are grouped to different rows. When the current SQL statement is not `COMMIT`, the `PREV_SAMPLE_TEXT` field is an empty string.

4.9.9.2 `events_statements_summary_by_digest_history`

The schema of the `events_statements_summary_by_digest_history` table is identical to that of the `events_statements_summary_by_digest` table. `events_statements_summary_by_digest_` ↪ stores the historical data used to troubleshoot anomalies or to compare monitoring metrics of different time.

The `SUMMARY_BEGIN_TIME` field and the `SUMMARY_END_TIME` field refer to the beginning time and the ending time of a historical period.

4.9.9.3 Troubleshooting examples

This section shows how to use the statement summary feature to troubleshoot SQL performance issues using two sample questions.

4.9.9.3.1 Could high SQL latency be caused by the server end?

In this example, the client shows slow performance with point queries on the employee table. You can perform a fuzzy search by SQL texts:

```
SELECT avg_latency, exec_count, query_sample_text
    FROM performance_schema.events_statements_summary_by_digest
    WHERE digest_text LIKE 'select * from employee%';
```

As shown in the result below, `avg_latency` of 1ms and 0.3ms are in the normal range. Therefore, it can be concluded that the server end is not the cause, and continue the troubleshooting with the client or the network.

```
+-------------+-----------+-----------------------------------------+
| avg_latency | exec_count | query_sample_text                      |
+-------------+-----------+-----------------------------------------+
|     1042040 |         2 | select * from employee where name='eric' |
|      345053 |         3 | select * from employee where id=3100 |
+-------------+-----------+-----------------------------------------+
2 rows in set (0.00 sec)
```

4.9.9.3.2 Which category of SQL statements c longest total time consumption?

If the QPS (queries/sec) decrease significantly from 10:00 A.M. to 10:30 A.M., you can find out the three categories of SQL statements with the longest time consumption from the history table:

```
SELECT sum_latency, avg_latency, exec_count, query_sample_text
    FROM performance_schema.events_statements_summary_by_digest_history
    WHERE summary_begin_time='2020-01-02 10:00:00'
    ORDER BY sum_latency DESC LIMIT 3;
```

The result shows that the following three SQL categories consume the longest time in total, which require focus on optimization.

```
+-------------+-------------+-----------+-----------------------------------------
   ↪
| sum_latency | avg_latency | exec_count | query_sample_text
   ↪                                        |
+-------------+-------------+-----------+-----------------------------------------
   ↪
|     7855660 |    1122237 |         7 | select avg(salary) from employee
   ↪ where company_id=2013       |
|     7241960 |    1448392 |         5 | select * from employee join company
   ↪ on employee.company_id=company.id |
|     2084081 |    1042040 |         2 | select * from employee where name='
   ↪ eric'                     |
+-------------+-------------+-----------+-----------------------------------------
   ↪
3 rows in set (0.00 sec)
```

4.9.9.4 Configurations

The statement summary feature is disabled by default. You can enable it by setting a system variable, for example:

```
set global tidb_enable_stmt_summary = true;
```

The statistics in the system table will be cleared if the statement summary feature is disabled, and will be re-calculated next time the statement summary feature is enabled. Tests have shown that enabling this feature has little impact on performance.

Another two system variables that control the statement summary:

- `tidb_stmt_summary_refresh_interval`: The interval at which the `events_statements_summary_b`
  ↪ table is refreshed. The time unit is second (s). The default value is `1800`.
- `tidb_stmt_summary_history_size`: The size of each SQL statement category historically stored in the `events_statements_summary_by_digest_history` table. The default value is `24`.

The following is a configuration example of statement summary:

```
set global tidb_stmt_summary_refresh_interval = 1800;
set global tidb_stmt_summary_history_size = 24;
```

When the above configuration takes effect, every 30 minutes the `events_statements_summary_by_dige`
↪ is cleared. `events_statements_summary_by_digest_history` stores data generated over the recent 12 hours.

The above two system variables have two scopes - global and session, which work a little differently from other system variables, as described below:

- Set the global variable to apply to the cluster immediately.
- Set the session variable to apply to the current TiDB server immediately. This is useful when you debug on a single TiDB server instance.
- The session variable has a higher read priority. The global variable will be read only if no session variable is set.
- Set the session variable to a blank string to re-read the global variable.

The statement summary tables are memory tables. To prevent potential memory issues, you need to limit the number of statements to be saved and the longest SQL display length. You can configure these limits using the following parameters under `[stmt-summary]` of `config.toml`:

- `max-stmt-count` limits the number of SQL statements that can be stored. The default value is `200`. If the set limit is exceeded, those SQL statements that recently remain unused will be cleared.
- `max-sql-length` specifies the longest display length of `DIGEST_TEXT` and `QUERY_SAMPLE_TEXT` ↪ . The default value is 4096.

> **Note:**
>
> The `tidb_stmt_summary_history_size`, `max-stmt-count`, and `max-sql-` ↪ `length` configuration items affect memory usage. It is recommended that you adjust these configurations based on your actual needs. It is not recommended to set them too large values.

4.9.9.5 Known limitations

The statement summary tables have the following limitations:

- Querying statement summary tables only returns the statement summary of the current TiDB server, not that of the entire cluster.
- The statement summary will be lost when the TiDB server restarts. This is because statement summary tables are memory tables, and the data is cached in memory instead of being persisted on storage.

4.9.10 Tune TiKV Performance

This document describes how to tune the TiKV parameters for optimal performance.

TiKV uses RocksDB for persistent storage at the bottom level of the TiKV architecture. Therefore, many of the performance parameters are related to RocksDB. TiKV uses two RocksDB instances: the default RocksDB instance stores KV data, the Raft RocksDB instance (RaftDB) stores Raft logs.

TiKV implements `Column Families` (CF) from RocksDB.

- The default RocksDB instance stores KV data in the `default`, `write` and `lock` CFs.
  - The `default` CF stores the actual data. The corresponding parameters are in `[rocksdb.defaultcf]`.
  - The `write` CF stores the version information in Multi-Version Concurrency Control (MVCC) and index-related data. The corresponding parameters are in `[rocksdb.writecf]`.
  - The `lock` CF stores the lock information. The system uses the default parameters.
- The Raft RocksDB (RaftDB) instance stores Raft logs.
  - The `default` CF stores the Raft log. The corresponding parameters are in `[raftdb.defaultcf]`.

After TiKV 3.0, by default, all CFs share one block cache instance. You can configure the size of the cache by setting the `capacity` parameter under `[storage.block-cache]`. The bigger the block cache, the more hot data can be cached, and the easier to read data, in the meantime, the more system memory is occupied. To use a separate block cache instance for each CF, set `shared=false` under `[storage.block-cache]`, and configure individual block cache size for each CF. For example, you can configure the size of `write` CF by setting the `block-cache-size` parameter under `[rocksdb.writecf]`.

Before TiKV 3.0, shared block cache is not supported, and you need to configure block cache for each CF individually.

Each CF also has a separate `write buffer`. You can configure the size by setting the `write-buffer-size` parameter.

4.9.10.1 Parameter specification

```
### Log level: trace, debug, warn, error, info, off.
log-level = "info"

[server]
### Set listening address
### addr = "127.0.0.1:20160"
```

```
### Size of thread pool for gRPC
### grpc-concurrency = 4
### The number of gRPC connections between each TiKV instance
### grpc-raft-conn-num = 1

### Most read requests from TiDB are sent to the coprocessor of TiKV. This
    ↪ parameter is used to set the number of threads
### of the coprocessor. If many read requests exist, add the number of
    ↪ threads and keep the number within that of the
### system CPU cores. For example, for a 32-core machine deployed with TiKV,
    ↪  you can even set this parameter to 30 in
### repeatable read scenarios. If this parameter is not set, TiKV
    ↪ automatically sets it to CPU cores * 0.8.
### end-point-concurrency = 8

### Tag the TiKV instances to schedule replicas.
### labels = {zone = "cn-east-1", host = "118", disk = "ssd"}

[storage]
### The data directory
### data-dir = "/tmp/tikv/store"

### In most cases, you can use the default value. When importing data, it is
    ↪  recommended to set the parameter to 1024000.
### scheduler-concurrency = 102400
### This parameter controls the number of write threads. When write
    ↪ operations occur frequently, set this parameter value
### higher. Run `top -H -p tikv-pid` and if the threads named `sched-worker-
    ↪ pool` are busy, set the value of parameter
### `scheduler-worker-pool-size` higher and increase the number of write
    ↪ threads.
### scheduler-worker-pool-size = 4

[storage.block-cache]
#### Whether to create a shared block cache for all RocksDB column families.
#### ## Block cache is used by RocksDB to cache uncompressed blocks. Big
    ↪ block cache can speed up read.
#### It is recommended to turn on shared block cache. Since only the total
    ↪ cache size need to be
#### set, it is easier to configure. In most cases, it should be able to
    ↪ auto-balance cache usage
#### between column families with standard LRU algorithm.
#### ## The rest of config in the storage.block-cache session is effective
    ↪ only when shared block cache
```

```
#### is on.
### shared = true

#### Size of the shared block cache. Normally it should be tuned to 30%-50%
   ↪ of system's total memory.
#### When the config is not set, it is decided by the sum of the following
   ↪ fields or their default
#### value:
####  * rocksdb.defaultcf.block-cache-size or 25% of system's total memory
####  * rocksdb.writecf.block-cache-size or 15% of system's total memory
####  * rocksdb.lockcf.block-cache-size or 2% of system's total memory
####  * raftdb.defaultcf.block-cache-size or 2% of system's total memory
#### ## To deploy multiple TiKV nodes on a single physical machine,
   ↪ configure this parameter explicitly.
#### Otherwise, the OOM problem might occur in TiKV.
### capacity = "1GB"

[pd]
### PD address
### endpoints = ["127.0.0.1:2379","127.0.0.2:2379","127.0.0.3:2379"]

[metric]
### The interval of pushing metrics to Prometheus Pushgateway
interval = "15s"
### Prometheus Pushgateway address
address = ""
job = "tikv"

[raftstore]
### The default value is true, which means writing the data on the disk
   ↪ compulsorily. If it is not in a business scenario
### of the financial security level, it is recommended to set the value to
   ↪ false to achieve better performance.
sync-log = true

### Raft RocksDB directory. The default value is Raft subdirectory of [
   ↪ storage.data-dir].
### If there are multiple disks on the machine, store the data of Raft
   ↪ RocksDB on different disks to improve TiKV performance.
### raftdb-path = "/tmp/tikv/store/raft"

region-max-size = "384MB"
### The threshold value of Region split
region-split-size = "256MB"
### When the data size change in a Region is larger than the threshold value
```

```
   ↪ , TiKV checks whether this Region needs split.
### To reduce the costs of scanning data in the checking process, set the
   ↪ value to 32MB during checking and set it to
### the default value in normal operation.
region-split-check-diff = "32MB"

[rocksdb]
### The maximum number of threads of RocksDB background tasks. The
   ↪ background tasks include compaction and flush.
### For detailed information why RocksDB needs to implement compaction, see
   ↪ RocksDB-related materials. When write
### traffic (like the importing data size) is big, it is recommended to
   ↪ enable more threads. But set the number of the enabled
### threads smaller than that of CPU cores. For example, when importing data
   ↪ , for a machine with a 32-core CPU,
### set the value to 28.
### max-background-jobs = 8

### The maximum number of file handles RocksDB can open
### max-open-files = 40960

### The file size limit of RocksDB MANIFEST. For more details, see https://
   ↪ github.com/facebook/rocksdb/wiki/MANIFEST
max-manifest-file-size = "20MB"

### The directory of RocksDB write-ahead logs. If there are two disks on the
   ↪  machine, store the RocksDB data and WAL logs
### on different disks to improve TiKV performance.
### wal-dir = "/tmp/tikv/store"

### Use the following two parameters to deal with RocksDB archiving WAL.
### For more details, see https://github.com/facebook/rocksdb/wiki/How-to-
   ↪ persist-in-memory-RocksDB-database%3F
### wal-ttl-seconds = 0
### wal-size-limit = 0

### In most cases, set the maximum total size of RocksDB WAL logs to the
   ↪ default value.
### max-total-wal-size = "4GB"

### Use this parameter to enable or disable the statistics of RocksDB.
### enable-statistics = true

### Use this parameter to enable the readahead feature during RocksDB
   ↪ compaction. If you are using mechanical disks, it is recommended to
```

```
   ↪ set the value to 2MB at least.
### compaction-readahead-size = "2MB"

[rocksdb.defaultcf]
### The data block size. RocksDB compresses data based on the unit of block.
### Similar to page in other databases, block is the smallest unit cached in
   ↪  block-cache.
block-size = "64KB"

### The compaction mode of each layer of RocksDB data. The optional values
   ↪ include no, snappy, zlib,
### bzip2, lz4, lz4hc, and zstd.
### "no:no:lz4:lz4:lz4:zstd:zstd" indicates there is no compaction of level0
   ↪  and level1; lz4 compaction algorithm is used
### from level2 to level4; zstd compaction algorithm is used from level5 to
   ↪ level6.
### "no" means no compaction. "lz4" is a compaction algorithm with moderate
   ↪ speed and compaction ratio. The
### compaction ratio of zlib is high. It is friendly to the storage space,
   ↪ but its compaction speed is slow. This
### compaction occupies many CPU resources. Different machines deploy
   ↪ compaction modes according to CPU and I/O resources.
### For example, if you use the compaction mode of "no:no:lz4:lz4:lz4:zstd:
   ↪ zstd" and find much I/O pressure of the
### system (run the iostat command to find %util lasts 100%, or run the top
   ↪ command to find many iowaits) when writing
### (importing) a lot of data while the CPU resources are adequate, you can
   ↪ compress level0 and level1 and exchange CPU
### resources for I/O resources. If you use the compaction mode of "no:no:
   ↪ lz4:lz4:lz4:zstd:zstd" and you find the I/O
### pressure of the system is not big when writing a lot of data, but CPU
   ↪ resources are inadequate. Then run the top
### command and choose the -H option. If you find a lot of bg threads (
   ↪ namely the compaction thread of RocksDB) are
### running, you can exchange I/O resources for CPU resources and change the
   ↪  compaction mode to "no:no:no:lz4:lz4:zstd:zstd".
### In a word, it aims at making full use of the existing resources of the
   ↪ system and improving TiKV performance
### in terms of the current resources.
compression-per-level = ["no", "no", "lz4", "lz4", "lz4", "zstd", "zstd"]

### The RocksDB memtable size
write-buffer-size = "128MB"

### The maximum number of the memtables. The data written into RocksDB is
```

```
   ↪ first recorded in the WAL log, and then inserted
### into memtables. When the memtable reaches the size limit of `write-
   ↪ buffer-size`, it turns into read only and generates
### a new memtable receiving new write operations. The flush threads of
   ↪ RocksDB will flush the read only memtable to the
### disks to become an sst file of level0. `max-background-flushes` controls
   ↪  the maximum number of flush threads. When the
### flush threads are busy, resulting in the number of the memtables waiting
   ↪  to be flushed to the disks reaching the limit
### of `max-write-buffer-number`, RocksDB stalls the new operation.
### "Stall" is a flow control mechanism of RocksDB. When importing data, you
   ↪  can set the `max-write-buffer-number` value
### higher, like 10.
max-write-buffer-number = 5

### When the number of sst files of level0 reaches the limit of `level0-
   ↪ slowdown-writes-trigger`, RocksDB
### tries to slow down the write operation, because too many sst files of
   ↪ level0 can cause higher read pressure of
### RocksDB. `level0-slowdown-writes-trigger` and `level0-stop-writes-
   ↪ trigger` are for the flow control of RocksDB.
### When the number of sst files of level0 reaches 4 (the default value),
   ↪ the sst files of level0 and the sst files
### of level1 which overlap those of level0 implement compaction to relieve
   ↪ the read pressure.
level0-slowdown-writes-trigger = 20

### When the number of sst files of level0 reaches the limit of `level0-stop
   ↪ -writes-trigger`, RocksDB stalls the new
### write operation.
level0-stop-writes-trigger = 36

### When the level1 data size reaches the limit value of `max-bytes-for-
   ↪ level-base`, the sst files of level1
### and their overlap sst files of level2 implement compaction. The golden
   ↪ rule: the first reference principle
### of setting `max-bytes-for-level-base` is guaranteeing that the `max-
   ↪ bytes-for-level-base` value is roughly equal to the
### data volume of level0. Thus unnecessary compaction is reduced. For
   ↪ example, if the compaction mode is
### "no:no:lz4:lz4:lz4:lz4:lz4", the `max-bytes-for-level-base` value is
   ↪ write-buffer-size * 4, because there is no
### compaction of level0 and level1 and the trigger condition of compaction
   ↪ for level0 is that the number of the
### sst files reaches 4 (the default value). When both level0 and level1
```

```
   ↪ adopt compaction, it is necessary to analyze
### RocksDB logs to know the size of an sst file compressed from an mentable
   ↪ . For example, if the file size is 32MB,
### the proposed value of `max-bytes-for-level-base` is 32MB * 4 = 128MB.
max-bytes-for-level-base = "512MB"

### The sst file size. The sst file size of level0 is influenced by the
   ↪ compaction algorithm of `write-buffer-size`
### and level0. `target-file-size-base` is used to control the size of a
   ↪ single sst file of level1-level6.
target-file-size-base = "32MB"

[rocksdb.writecf]
### Set it the same as `rocksdb.defaultcf.compression-per-level`.
compression-per-level = ["no", "no", "lz4", "lz4", "lz4", "zstd", "zstd"]

### Set it the same as `rocksdb.defaultcf.write-buffer-size`.
write-buffer-size = "128MB"
max-write-buffer-number = 5
min-write-buffer-number-to-merge = 1

### Set it the same as `rocksdb.defaultcf.max-bytes-for-level-base`.
max-bytes-for-level-base = "512MB"
target-file-size-base = "32MB"

[raftdb]
### The maximum number of the file handles RaftDB can open
### max-open-files = 40960

### Configure this parameter to enable or disable the RaftDB statistics
   ↪ information.
### enable-statistics = true

### Enable the readahead feature in RaftDB compaction. If you are using
   ↪ mechanical disks, it is recommended to set
### this value to 2MB at least.
### compaction-readahead-size = "2MB"

[raftdb.defaultcf]
### Set it the same as `rocksdb.defaultcf.compression-per-level`.
compression-per-level = ["no", "no", "lz4", "lz4", "lz4", "zstd", "zstd"]

### Set it the same as `rocksdb.defaultcf.write-buffer-size`.
write-buffer-size = "128MB"
max-write-buffer-number = 5
```

```
min-write-buffer-number-to-merge = 1

### Set it the same as `rocksdb.defaultcf.max-bytes-for-level-base`.
max-bytes-for-level-base = "512MB"
target-file-size-base = "32MB"
```

4.9.10.2 TiKV memory usage

Besides `block cache` and `write buffer` which occupy the system memory, the system memory is occupied in the following scenarios:

- Some of the memory is reserved as the system's page cache.
- When TiKV processes large queries such as `select * from ...`, it reads data, generates the corresponding data structure in the memory, and returns this structure to TiDB. During this process, TiKV occupies some of the memory.

4.9.10.3 Recommended configuration of TiKV

- In production environments, it is not recommended to deploy TiKV on the machine whose CPU cores are less than 8 or the memory is less than 32GB.
- If you demand a high write throughput, it is recommended to use a disk with good throughput capacity.
- If you demand a very low read-write latency, it is recommended to use SSD with high IOPS.

4.9.11 Operating System Tuning

This document introduces how to tune each subsystem of CentOS 7.

Note:

- The default configuration of the CentOS 7 operating system is suitable for most services running under moderate workloads. Adjusting the performance of a particular subsystem might negatively affects other subsystems. Therefore, before tuning the system, back up all the user data and configuration information.
- Fully test all the changes in the test environment before applying them to the production environment.

4.9.11.1 Performance analysis methods

System tuning must be based on the results of system performance analysis. This section lists common methods for performance analysis.

4.9.11.1.1 In 60 seconds

Linux Performance Analysis in 60,000 Milliseconds is published by the author Brendan Gregg and the Netflix Performance Engineering team. All tools used can be obtained from the official release of Linux. You can analyze outputs of the following list items to troubleshoot most common performance issues.

- `uptime`
- `dmesg | tail`
- `vmstat 1`
- `mpstat -P ALL 1`
- `pidstat 1`
- `iostat -xz 1`
- `free -m`
- `sar -n DEV 1`
- `sar -n TCP,ETCP 1`
- `top`

For detailed usage, see the corresponding `man` instructions.

4.9.11.1.2 perf

perf is an important performance analysis tool provided by the Linux kernel, which covers hardware level (CPU/PMU, performance monitoring unit) features and software features (software counters, trace points). For detailed usage, see perf Examples.

4.9.11.1.3 BCC/bpftrace

Starting from CentOS 7.6, the Linux kernel has supported Berkeley Packet Filter (BPF). Therefore, you can choose proper tools to conduct an in-depth analysis based on the results in In 60 seconds. Compared with perf/ftrace, BPF provides programmability and smaller performance overhead. Compared with kprobe, BPF provides higher security and is more suitable for the production environments. For detailed usage of the BCC toolkit, see BPF Compiler Collection (BCC).

4.9.11.2 Performance tuning

This section introduces performance tuning based on the classified kernel subsystems.

4.9.11.2.1 CPU—frequency scaling

cpufreq is a module that dynamically adjusts the CPU frequency. It supports five modes. To ensure service performance, select the performance mode and fix the CPU frequency at the highest supported operating frequency without dynamic adjustment. The command for this operation is `cpupower frequency-set --governor performance`.

4.9.11.2.2 CPU—interrupt affinity

- Automatic balance can be implemented through the `irqbalance` service.
- Manual balance:
  - Identify the devices that need to balance interrupts. Starting from CentOS 7.5, the system automatically configures the best interrupt affinity for certain devices and their drivers, such as devices that use the `be2iscsi` driver and NVMe settings. You can no longer manually configure interrupt affinity for such devices.
  - For other devices, check the chip manual to see whether these devices support distributing interrupts.
    * If they do not, all interrupts of these devices are routed to the same CPU and cannot be modified.
    * If they do, calculate the `smp_affinity` mask and set the corresponding configuration file. For details, see the kernel document.

4.9.11.2.3 NUMA CPU binding

To avoid accessing memory across Non-Uniform Memory Access (NUMA) nodes as much as possible, you can bind a thread/process to certain CPU cores by setting the CPU affinity of the thread. For ordinary programs, you can use the `numactl` command for the CPU binding. For detailed usage, see the Linux manual pages. For network interface card (NIC) interrupts, see tune network.

4.9.11.2.4 Memory—transparent huge page (THP)

It is **NOT** recommended to use THP for database applications, because databases often have sparse rather than continuous memory access patterns. If high-level memory fragmentation is serious, a higher latency will occur when THP pages are allocated. If the direct compaction is enabled for THP, the CPU usage will surge. Therefore, it is recommended to disable THP.

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

4.9.11.2.5 Memory—virtual memory parameters

- `dirty_ratio` percentage ratio. When the total amount of dirty page caches reach this percentage ratio of the total system memory, the system starts to use the `pdflush` operation to write the dirty page caches to disk. The default value of `dirty_ratio` is 20% and usually does not need adjustment. For high-performance SSDs such as NVMe devices, lowering this value helps improve the efficiency of memory reclamation.
- `dirty_background_ratio` percentage ratio. When the total amount of dirty page caches reach this percentage ratio of the total system memory, the system starts to write the dirty page caches to the disk in the background. The default value of `dirty_ratio` is 10% and usually does not need adjustment. For high-performance SSDs such as NVMe devices, setting a lower value helps improve the efficiency of memory reclamation.

4.9.11.2.6 Storage and file system

The core I/O stack link is long, including the file system layer, the block device layer, and the driver layer.

I/O scheduler

The I/O scheduler determines when and how long I/O operations run on the storage device. It is also called I/O elevator. For SSD devices, it is recommended to set the I/O scheduling policy to noop.

```
echo noop > /sys/block/${SSD_DEV_NAME}/queue/scheduler
```

Formatting parameters—block size

Blocks are the working units of the file system. The block size determines how much data can be stored in a single block, and thus determines the minimum amount of data to be written or read each time.

The default block size is suitable for most scenarios. However, if the block size (or the size of multiple blocks) is the same or slightly larger than the amount of data normally read or written each time, the file system performs better and the data storage efficiency is higher. Small files still use the entire block. Files can be distributed among multiple blocks, but this will increase runtime overhead.

When using the `mkfs` command to format a device, specify the block size as a part of the file system options. The parameters that specify the block size vary with the file system. For details, see the corresponding `mkfs` manual pages, such as using `man mkfs.ext4`.

`mount` parameters

If the `noatime` option is enabled in the `mount` command, the update of metadata is disabled when files are read. If the `nodiratime` behavior is enabled, the update of metadata is disabled when the directory is read.

4.9.11.2.7 Network tuning

The network subsystem consists of many different parts with sensitive connections. The CentOS 7 network subsystem is designed to provide the best performance for most workloads and automatically optimizes the performance of these workloads. Therefore, usually you do not need to manually adjust network performance.

Network issues are usually caused by issues of hardware or related devices. So before tuning the protocol stack, rule out hardware issues.

Although the network stack is largely self-optimizing, the following aspects in the network packet processing might become the bottleneck and affect performance:

- NIC hardware cache: To correctly observe the packet loss at the hardware level, use the `ethtool -S ${NIC_DEV_NAME}` command to observe the `drops` field. When packet loss occurs, it might be that the processing speed of the hard/soft interrupts cannot catch up with the receiving speed of NIC. If the received buffer size is less than the upper limit, you can also try to increase the RX buffer to avoid packet loss. The query command is: `ethtool -g ${NIC_DEV_NAME}`, and the modification command is `ethtool -G ${NIC_DEV_NAME}`.

- Hardware interrupts: If the NIC supports the Receive-Side Scaling (RSS, also called multi-NIC receiving) feature, observe the `/proc/interrupts` NIC interrupts. If the interrupts are uneven, see CPU—frequency scaling, CPU—interrupt affinity, and NUMA CPU binding. If the NIC does not support RSS or the number of RSS is much smaller than the number of physical CPU cores, configure Receive Packet Steering (RPS, which can be regarded as the software implementation of RSS), and the RPS extension Receive Flow Steering (RFS). For detailed configuration, see the kernel document.

- Software interrupts: Observe the monitoring of `/proc/net/softnet_stat`. If the values of the other columns except the third column are increasing, properly adjust the value of `net.core.netdev_budget` or `net.core.dev_weight` for `softirq` to get more CPU time. In addition, you also need to check the CPU usage to determine which tasks are frequently using the CPU and whether they can be optimized.

- Receive queue of application sockets: Monitor the `Resv-q` column of `ss -nmp`. If the queue is full, consider increasing the size of the application socket cache or use the automatic cache adjustment method. In addition, consider whether you can optimize the architecture of the application layer and reduce the interval between reading sockets.

- Ethernet flow control: If the NIC and switch support the flow control feature, you can use this feature to leave some time for the kernel to process the data in the NIC queue, to avoid the issue of NIC buffer overflow.

- Interrupts coalescing: Too frequent hardware interrupts reduces system performance, and too late hardware interrupts causes packet loss. Newer NICs support the interrupt coalescing feature and allow the driver to automatically adjust the number of hardware interrupts. You can execute `ethtool -c ${NIC_DEV_NAME}` to check and `ethtool -C ${NIC_DEV_NAME}` to enable this feature. The adaptive mode allows the NIC

to automatically adjust the interrupt coalescing. In this mode, the driver checks the traffic mode and kernel receiving mode, and evaluates the coalescing settings in real time to prevent packet loss. NICs of different brands have different features and default configurations. For details, see the NIC manuals.

- Adapter queue: Before processing the protocol stack, the kernel uses this queue to buffer the data received by the NIC, and each CPU has its own backlog queue. The maximum number of packets that can be cached in this queue is `netdev_max_backlog` ↪ . Observe the second column of `/proc/net/softnet_stat`. When the second column of a row continues to increase, it means that the CPU [row-1] queue is full and the data packet is lost. To resolve this problem, continue to double the `net.core.` ↪ `netdev_max_backlog` value.
- Send queue: The length value of a send queue determines the number of packets that can be queued before sending. The default value is `1000`, which is sufficient for 10 Gbps. But if you have observed the value of TX errors from the output of `ip -s link`, you can try to double it: `ip link set dev ${NIC_DEV_NAME} txqueuelen 2000`.
- Driver: NIC drivers usually provide tuning parameters. See the device hardware manual and its driver documentation.

4.9.12 Column Pruning

The basic idea of column pruning is that for columns not used in the operator, the optimizer does not need to retain them during optimization. Removing these columns reduces the use of I/O resources and facilitates the subsequent optimization. The following is an example of column repetition:

Suppose there are four columns (a, b, c, and d) in table t. You can execute the following statement:

```
select a from t where b> 5
```

In this query, only column a and column b are used, and column c and column d are redundant. Regarding the query plan of this statement, the `Selection` operator uses column b. Then the `DataSource` operator uses columns a and column b. Columns c and column d can be pruned because the `DataSource` operator does not read them.

Therefore, when TiDB performs a top-down scanning during the logic optimization phase, redundant columns are pruned to reduce waste of resources. This scanning process is called "Column Pruning", corresponding to the `columnPruner` rule. If you want to disable this rule, refer to The Blocklist of Optimization Rules and Expression Pushdown.

4.10 Key Monitoring Metrics

4.10.1 Key Metrics

If you use TiDB Ansible to deploy the TiDB cluster, the monitoring system is deployed at the same time. For more information, see TiDB Monitoring Framework Overview.

The Grafana dashboard is divided into a series of sub dashboards which include Overview, PD, TiDB, TiKV, Node_exporter, Disk Performance, and so on. A lot of metrics are there to help you diagnose.

For routine operations, you can get an overview of the component (PD, TiDB, TiKV) status and the entire cluster from the Overview dashboard, where the key metrics are displayed. This document provides a detailed description of these key metrics.

4.10.1.1 Key metrics description

To understand the key metrics displayed on the Overview dashboard, check the following table:

Service	Panel Name	Description	Normal Range
Services Port Status	Services Online	the online nodes number of each service	
Services Port Status	Services Offline	the offline nodes number of each service	
PD	Storage Capacity	the total storage capacity of the TiDB cluster	
PD	Current Storage Size	the occupied storage capacity of the TiDB cluster	
PD	Number of Regions	the total number of Regions of the current cluster	
PD	Leader Balance Ratio	the leader ratio difference of the nodes with the biggest leader ratio and the smallest leader ratio	It is less than 5% for a balanced situation and becomes bigger when you restart a node.

Service	Panel Name	Description	Normal Range
PD	Region Balance Ratio	the Region ratio difference of the nodes with the biggest Region ratio and the smallest Region ratio	It is less than 5% for a balanced situation and becomes bigger when you add or remove a node.
PD	Store Status – Up Stores	the number of TiKV nodes that are up	
PD	Store Status – Disconnect Stores	the number of TiKV nodes that encounter abnormal communication within a short time	
PD	Store Status – LowSpace Stores	the number of TiKV nodes with an available space of less than 20%	
PD	Store Status – Down Stores	the number of TiKV nodes that are down	The normal value is `0`. If the number is bigger than `0`, it means some node(s) are abnormal.
PD	Store Status – Offline Stores	the number of TiKV nodes (still providing service) that are being made offline	
PD	Store Status – Tombstone Stores	the number of TiKV nodes that are successfully offline	
PD	99% completed_cmds_duration_seconds	the 99th percentile duration to complete a pd-server request	less than 5ms
PD	handle_requests_duration_seconds	the request duration of a PD request	
PD	Region health	The state of each Region.	Generally, the number of pending peers is less than 100, and that of the missing peers cannot always be greater than `0`.

Service	Panel Name	Description	Normal Range
PD	Hot write Region's leader distribution	The total number of leaders who are the write hotspots on each TiKV instance.	
PD	Hot read Region's leader distribution	The total number of leaders who are the read hotspots on each TiKV instance.	
PD	Region heartbeat report	The count of heartbeats reported to PD per instance.	
PD	99% Region heartbeat latency	The heartbeat latency per TiKV instance (P99).	
TiDB	Statement OPS	the total number of executed SQL statements, including `SELECT`, `INSERT`, `UPDATE` and so on	
TiDB	Duration	the execution time of a SQL statement	
TiDB	QPS By Instance	the QPS on each TiDB instance	
TiDB	Failed Query OPM	the number of failed SQL statements, including syntax error and key conflicts and so on	
TiDB	Connection Count	the connection number of each TiDB instance	
TiDB	Heap Memory Usage	the size of heap memory used by each TiDB instance	
TiDB	Transaction OPS	the number of executed transactions per second	
TiDB	Transaction Duration	the execution time of a transaction	
TiDB	KV Cmd OPS	the number of executed KV commands	
TiDB	KV Cmd Duration 99	the execution time of the KV command	
TiDB	PD TSO OPS	the number of TSO that TiDB obtains from PD	
TiDB	PD TSO Wait Duration	the time consumed when TiDB obtains TSO from PD	
TiDB	TiClient Region Error OPS	the number of Region related errors returned by TiKV	
TiDB	Lock Resolve OPS	the number of transaction related conflicts	
TiDB	Load Schema Duration	the time consumed when TiDB obtains Schema from TiKV	

Service	Panel Name	Description	Normal Range
TiDB	KV Backoff OPS	the number of errors returned by TiKV (such as transaction conflicts)	
TiKV	leader	the number of leaders on each TiKV node	
TiKV	region	the number of Regions on each TiKV node	
TiKV	CPU	the CPU usage ratio on each TiKV node	
TiKV	Memory	the memory usage on each TiKV node	
TiKV	store size	the data amount on each TiKV node	
TiKV	cf size	the data amount on different CFs in the cluster	
TiKV	channel full	`No data points` is displayed in normal conditions. If a monitoring value displays, it means the corresponding TiKV node fails to handle the messages	
TiKV	server report failures	`No data points` is displayed in normal conditions. If `Unreachable` is displayed, it means TiKV encounters a communication issue.	
TiKV	scheduler pending commands	the number of commits on queue	Occasional value peaks are normal.
TiKV	coprocessor pending requests	the number of requests on queue	`0` or very small
TiKV	coprocessor executor count	the number of various query operations	
TiKV	coprocessor request duration	the time consumed by TiKV queries	

Service	Panel Name	Description	Normal Range
TiKV	raft store CPU	the CPU usage ratio of the raftstore thread	The default number of threads is 2 (configured by `raftstore.store-pool-size`). A value of over 80% for a single thread indicates that the CPU usage ratio is very high.
TiKV	Coprocessor CPU	the CPU usage ratio of the TiKV query thread, related to the application; complex queries consume a great deal of CPU	
System Info	Vcores	the number of CPU cores	
System Info	Memory	the total memory	
System Info	CPU Usage	the CPU usage ratio, 100% at a maximum	
System Info	Load [1m]	the overload within 1 minute	
System Info	Memory Available	the size of the available memory	
System Info	Network Traffic	the statistics of the network traffic	
System Info	TCP Retrans	the statistics about network monitoring and TCP	
System Info	IO Util	the disk usage ratio, 100% at a maximum; generally you need to consider adding a new node when the usage ratio is up to 80% ~ 90%	

4.10.1.2 Interface of the Overview dashboard

Figure 268: Overview Dashboard

4.10.2 TiDB Monitoring Metrics

If you use TiDB Ansible to deploy the TiDB cluster, the monitoring system is deployed at the same time. For more information, see TiDB Monitoring Framework Overview.

The Grafana dashboard is divided into a series of sub dashboards which include Overview, PD, TiDB, TiKV, Node_exporter, Disk Performance, and so on. The TiDB dashboard consists of the TiDB panel and the TiDB Summary panel, and what displays on the TiDB Summary panel also displays on the TiDB panel. A lot of metrics are there to help you diagnose.

This document describes some key monitoring metrics displayed on the TiDB dashboard.

4.10.2.1 Key metrics description

To understand the key metrics displayed on the TiDB dashboard, check the following list:

- Query Summary
  - Duration: the execution time of a SQL statement
  - QPS: the statistics of OKs and Errors according to the SQL execution result on each TiDB instance
  - Statement OPS: the statistics of executed SQL statements (including `SELECT`, `INSERT`, `UPDATE` and so on)
  - QPS By Instance: the QPS on each TiDB instance
  - Failed Query OPM: the statistics of error types (such as syntax errors and primary key conflicts) according to the errors happening when executing SQL statements on each TiDB instance
  - Slow query: the statistics of the processing time of slow queries (the time cost of the entire slow query, the time cost of Coprocessor`,` and the waiting time for Coprocessor scheduling)
  - 999/99/95/80 Duration: the statistics of the execution time for different types of SQL statements (different percentiles)
- Query Detail
  - Duration 80/95/99/999 By Instance: the statistics of the execution time for SQL statements on each TiDB instance (different percentiles)
  - Failed Query OPM Detail: the statistics of error types (such as syntax errors and primary key conflicts) according to the errors happening when executing SQL statements in the entire cluster
  - Internal SQL OPS: the statistics of the executed SQL statements within TiDB
- Server
  - Uptime: the runtime of TiDB
  - Memory Usage: the statistics of memory usage of different TiDB instances
  - CPU Usage: the statistics of CPU usage of different TiDB instances
  - Connection Count: the number of clients connected to each TiDB instance
  - Open FD Count: the statistics of opened file descriptors of different TiDB instances
  - Goroutine Count: the number of Goroutines of different TiDB instances
  - Go GC Duration: the statistics of GC time of different TiDB instances
  - Go Threads: the number of threads of different TiDB instances
  - Go GC Count: the number of times that GC is executed on different TiDB instances
  - Go GC CPU Usage: the statistics of GC CPU usage of different TiDB instances
  - Events OPM: the statistics of key events, such as “start”, “close”, “graceful-shutdown”,“kill”, “hang”, and so on
  - Keep Alive OPM: the number of times that the metrics are refreshed every minute on different TiDB instances
  - Prepare Statement Count: the number of `Prepare` statements that are executed on each TiDB instance and the total count of them
  - Time Jump Back OPS: the number of times that the time rewinds every second on different TiDB instances

  - Heap Memory Usage: the heap memory size used by each TiDB instance
  - Uncommon Error OPM: the statistics of abnormal TiDB errors, including panic, binlog write failure, and so on
  - Handshake Error OPS: the number of times that a handshake error occurs every second on different TiDB instances
  - Get Token Duration: the time cost of getting Token after establishing the connection

- Transaction

  - Transaction OPS: the statistics of executed transactions
  - Duration: the execution time of a transaction
  - Transaction Retry Num: the number of times that a transaction retries
  - Transaction Statement Num: the number of SQL statements in a transaction
  - Session Retry Error OPS: the number of errors encountered during the transaction retry
  - Local Latch Wait Duration: the waiting time of a local transaction

- Executor

  - Parse Duration: the statistics of the parsing time of SQL statements
  - Compile Duration: the statistics of the time of compiling an SQL AST to the execution plan
  - Execution Duration: the statistics of the execution time for SQL statements
  - Expensive Executor OPS: the statistics of the operators that consume many system resources, including `Merge Join`, `Hash Join`, `Index Look Up Join`, `Hash Agg`, `Stream Agg`, `Sort`, `TopN`, and so on
  - Queries Using Plan Cache OPS: the statistics of queries using the Plan Cache

- Distsql

  - Distsql Duration: the processing time of Distsql statements
  - Distsql QPS: the statistics of Distsql statements
  - Distsql Partial QPS: the number of Partial results every second
  - Scan Keys Num: the number of keys that each query scans
  - Scan Keys Partial Num: the number of keys that each Partial result scans
  - Partial Num: the number of Partial results for each SQL statement

- KV Errors

  - KV Retry Duration: the time that a KV retry request lasts
  - TiClient Region Error OPS: the number of Region related error messages returned by TiKV
  - KV Backoff OPS: the number of error messages (transaction conflicts and so on) returned by TiKV
  - Lock Resolve OPS: the number of errors related to transaction conflicts
  - Other Errors OPS: the number of other types of errors, including clearing locks and updating SafePoint

- KV Duration
  - KV Request Duration 999 by store: the execution time of a KV request, displayed according to TiKV
  - KV Request Duration 999 by type: the execution time of a KV request, displayed according to the request type
  - KV Cmd Duration 99/999: the execution time of KV commands
- KV Count
  - KV Cmd OPS: the statistics of executed KV commands
  - KV Txn OPS: the statistics of started transactions
  - Txn Regions Num 90: the statistics of Regions used by the transaction
  - Txn Write Size Bytes 100: the statistics of bytes written by the transaction
  - Txn Write KV Num 100: the statistics of KVs written by the transaction
  - Load SafePoint OPS: the statistics of operations that update SafePoint
- PD Client
  - PD Client CMD OPS: the statistics of commands executed by PD Client
  - PD Client CMD Duration: the time it takes PD Client to execute commands
  - PD Client CMD Fail OPS: the statistics of failed commands executed by PD Client
  - PD TSO OPS: the number of TSO that TiDB obtains from PD
  - PD TSO Wait Duration: the time it takes TiDB to obtain TSO from PD
  - PD TSO RPC Duration: the time it takes TiDB to obtain TSO gRPC interface from PD
- Schema Load
  - Load Schema Duration: the time it takes TiDB to obtain the schema from TiKV
  - Load Schema OPS: the statistics of the schemas that TiDB obtains from TiKV
  - Schema Lease Error OPM: the Schema Lease error, including two types named "change" and "outdate"; an alarm is triggered when an "outdate" error occurs
- DDL
  - DDL Duration 95: the statistics of DDL statements processing time
  - Batch Add Index Duration 100: the statistics of the time that it takes each Batch to create the index
  - DDL Waiting Jobs Count: the number of DDL tasks that are waiting
  - DDL META OPM: the number of times that a DDL obtains META every minute
  - Deploy Syncer Duration: the time consumed by Schema Version Syncer initialization, restart, and clearing up operations
  - Owner Handle Syncer Duration: the time that it takes the DDL Owner to update, obtain, and check the Schema Version
  - Update Self Version Duration: the time consumed by updating the version information of Schema Version Syncer
- Statistics

  - Auto Analyze Duration 95: the time consumed by automatic `ANALYZE`
  - Auto Analyze QPS: the statistics of automatic `ANALYZE`
  - Stats Inaccuracy Rate: the information of the statistics inaccuracy rate
  - Pseudo Estimation OPS: the number of the SQL statements optimized using pseudo statistics
  - Dump Feedback OPS: the number of stored statistical Feedbacks
  - Update Stats OPS: the statistics of using Feedback to update the statistics information
  - Significant Feedback: the number of significant Feedback pieces that update the statistics information

- Meta

  - AutoID QPS: AutoID related statistics, including three operations (global ID allocation, a single table AutoID allocation, a single table AutoID Rebase)
  - AutoID Duration: the time consumed by AutoID related operations
  - Meta Operations Duration 99: the latency of Meta operations

- GC

  - Worker Action OPM: the statistics of GC related operations, including `run_job`, `resolve_lock`, and `delete\_range`
  - Duration 99: the time consumed by GC related operations
  - GC Failure OPM: the number of failed GC related operations
  - Action Result OPM: the number of results of GC-related operations
  - Too Many Locks Error OPM: the number of the error that GC clears up too many locks

- Batch Client

  - Pending Request Count by TiKV: the number of Batch messages that are pending processing
  - Wait Duration 95: the waiting time of Batch messages that are pending processing
  - Batch Client Unavailable Duration 95: the unavailable time of the Batch client

4.10.3 Key Monitoring Metrics of PD

If you use TiDB Ansible to deploy the TiDB cluster, the monitoring system is deployed at the same time. For more information, see Overview of the Monitoring Framework.

The Grafana dashboard is divided into a series of sub dashboards which include Overview, PD, TiDB, TiKV, Node_exporter, Disk Performance, and so on. A lot of metrics are there to help you diagnose.

You can get an overview of the component PD status from the PD dashboard, where the key metrics are displayed. This document provides a detailed description of these key metrics.

4.10.3.1 Key metrics description

To understand the key metrics displayed on the Overview dashboard, check the following table:

Service	Panel name	Description	Normal range
Cluster	PD role	The role of the current PD	
Cluster	Storage capacity	The total capacity size of the cluster	
Cluster	Current storage size	The current storage size of the cluster	
Cluster	Current storage usage	The total number of Regions without replicas	
Cluster	Normal stores	The count of healthy stores	
Cluster	Abnormal stores	The count of unhealthy stores	The normal value is `0`. If the number is bigger than `0`, it means at least one instance is abnormal.
Cluster	Current peer count	The current peer count of the cluster	
Cluster	Number of Regions	The total number of Regions of the cluster	
Cluster	PD scheduler config	The list of PD scheduler configurations	
Cluster	Region label isolation level	The number of Regions in different label levels	
Cluster	Label distribution	The distribution status of the labels in the cluster	
Cluster	pd_cluster_metadata	The metadata of the PD cluster including cluster ID, the timestamp, and the generated ID.	
Cluster	Region health	The health status of Regions indicated via count of unusual Regions including pending peers, down peers, extra peers, offline peers, missing peers, learner peers and incorrect namespaces	The number of pending peers should be less than `100`. The missing peers should not be persistently greater than `0`.

Service	Panel name	Description	Normal range
Statistics - Balance	Store capacity	The capacity size per TiKV instance	
Statistics - Balance	Store available	The available capacity size per TiKV instance	
Statistics - Balance	Store used	The used capacity size per TiKV instance	
Statistics - Balance	Size amplification	The size amplification ratio per TiKV instance, which is equal to (Store Region size)/(Store used capacity size)	
Statistics - Balance	Size available ratio	The size availability ratio per TiKV instance, which is equal to (Store available capacity size)/(Store capacity size)	
Statistics - Balance	Store leader score	The leader score per TiKV instance	
Statistics - Balance	Store Region score	The Region score per TiKV instance	
Statistics - Balance	Store leader size	The total leader size per TiKV instance	
Statistics - Balance	Store Region size	The total Region size per TiKV instance	
Statistics - Balance	Store leader count	The leader count per TiKV instance	
Statistics - Balance	Store Region count	The Region count per TiKV instance	
Statistics - Hotspot	Leader distribution in hot write Regions	The total number of leader Regions in hot write on each TiKV instance	
Statistics - Hotspot	Peer distribution in hot write Regions	The total number of peer Regions under in hot write on each TiKV instance	
Statistics - Hotspot	Leader written bytes in hot write Regions	The total written bytes by Leader regions in hot write on leader Regions for each TiKV instance	
Statistics - Hotspot	Peer written bytes in hot write Regions	The total bytes of hot write on peer Regions per each TiKV instance	

Service	Panel name	Description	Normal range
Statistics - Hotspot	Leader distribution in hot read Regions	The total number of leader Regions in hot read per each TiKV instance	
Statistics - Hotspot	Peer distribution in hot read Regions	The total number of Regions which are not leader under hot read per each TiKV instance	
Statistics - Hotspot	Leader read bytes in hot read Regions	The total bytes of hot read on leader Regions per each TiKV instance	
Statistics - Hotspot	Peer read bytes in hot read Regions	The total bytes of hot read on peer Regions per TiKV instance	
Scheduler	Running schedulers	The current running schedulers	
Scheduler	Balance leader movement	The leader movement details among TiKV instances	
Scheduler	Balance Region movement	The Region movement details among TiKV instances	
Scheduler	Balance leader event	The count of balance leader events	
Scheduler	Balance Region event	The count of balance Region events	
Scheduler	Balance leader scheduler	The inner status of balance leader scheduler	
Scheduler	Balance Region scheduler	The inner status of balance Region scheduler	
Scheduler	Namespace checker	The namespace checker's status	
Scheduler	Replica checker	The replica checker's status	
Scheduler	Region merge checker	The merge checker's status	
Operator	Schedule operator create	The number of newly created operators per type	
Operator	Schedule operator check	The number of checked operator per type. It mainly checks if the current step is finished; if yes, it returns the next step to be executed.	
Operator	Schedule operator finish	The number of finished operators per type	

Service	Panel name	Description	Normal range
Operator	Schedule operator timeout	The number of timeout operators per type	
Operator	Schedule operator replaced or canceled	The number of replaced or canceled operators per type	
Operator	Schedule operators count by state	The number of operators per state	
Operator	99% Operator finish duration	The operator step duration (P99)	
Operator	50% Operator finish duration	The operator duration (P50)	
Operator	99% Operator step duration	The operator step duration (P99)	
Operator	50% Operator step duration	The operator step duration (P50)	
gRPC	Completed commands rate	The rate per command type type at which gRPC commands are completed	
gRPC	99% Completed commands duration	The rate per command type type at which gRPC commands are completed (P99)	
etcd	Transaction handling rate	The rate at which etcd handles transactions	
etcd	99% transactions duration	The transaction handling rate (P99)	
etcd	99% WAL fsync duration	The time consumed for writing WAL into the persistent storage (P99)	The value is less than `1s`.
etcd	99% Peer round trip time seconds	The network latency for etcd (P99)	The value is less than `1s`.
etcd	etcd disk wal fsync rate	The rate of writing WAL into the persistent storage	
etcd	Raft term	The current term of Raft	
etcd	Raft committed index	The last committed index of Raft	
etcd	Raft applied index	The last applied index of Raft	

Service	Panel name	Description	Normal range
TiDB	Handled requests count	The count of TiDB requests	
TiDB	Request handling duration	The time consumed for handling TiDB requests	It should be less than `100ms` (P99).
Heartbeat	Region heartbeat report	The count of heartbeats reported reported to PD per instance	
Heartbeat	Region heartbeat report error	The count of heartbeats with the `error` status	
Heartbeat	Region heartbeat report active	The count of heartbeats with the `ok` status	
Heartbeat	Region schedule push	The count of corresponding schedule commands sent from PD per TiKV instance	
Heartbeat	99% Region heartbeat latency	The heartbeat latency per TiKV instance (P99)	
Region storage	Syncer index	The maximum index in the Region change history recorded by the leader	
Region storage	History last index	The last index where the Region change history is synchronized	successfully with the follower

4.10.3.2 PD dashboard interface

4.10.3.2.1 Cluster

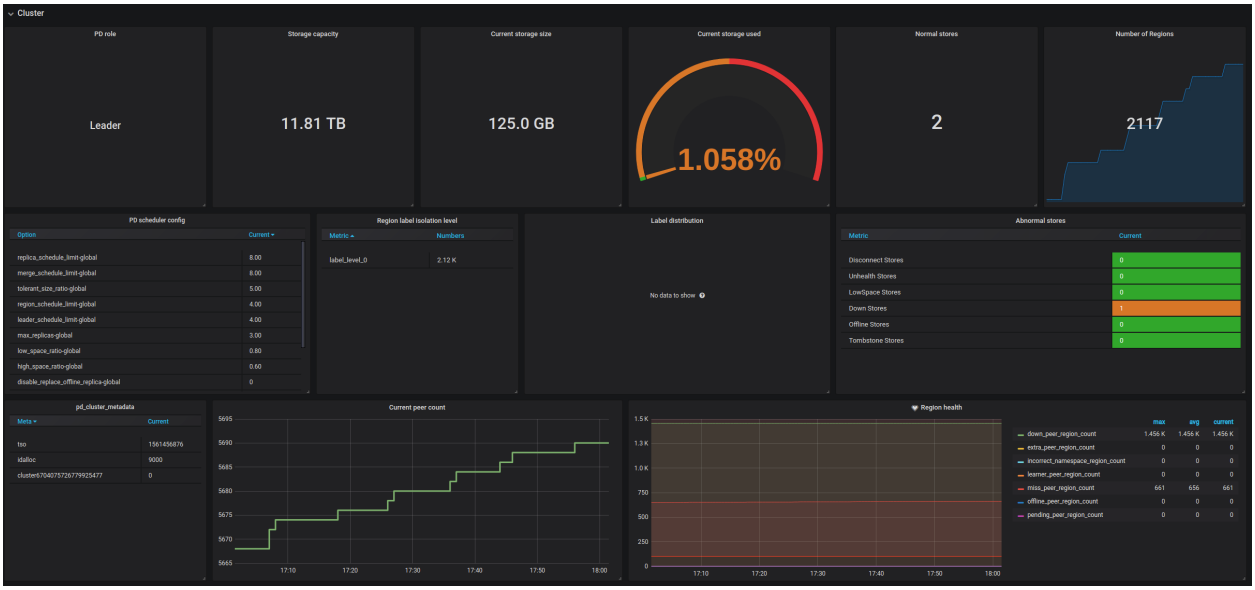

Figure 269: PD Dashboard - Cluster metrics

4.10.3.2.2 Statistics - Balance

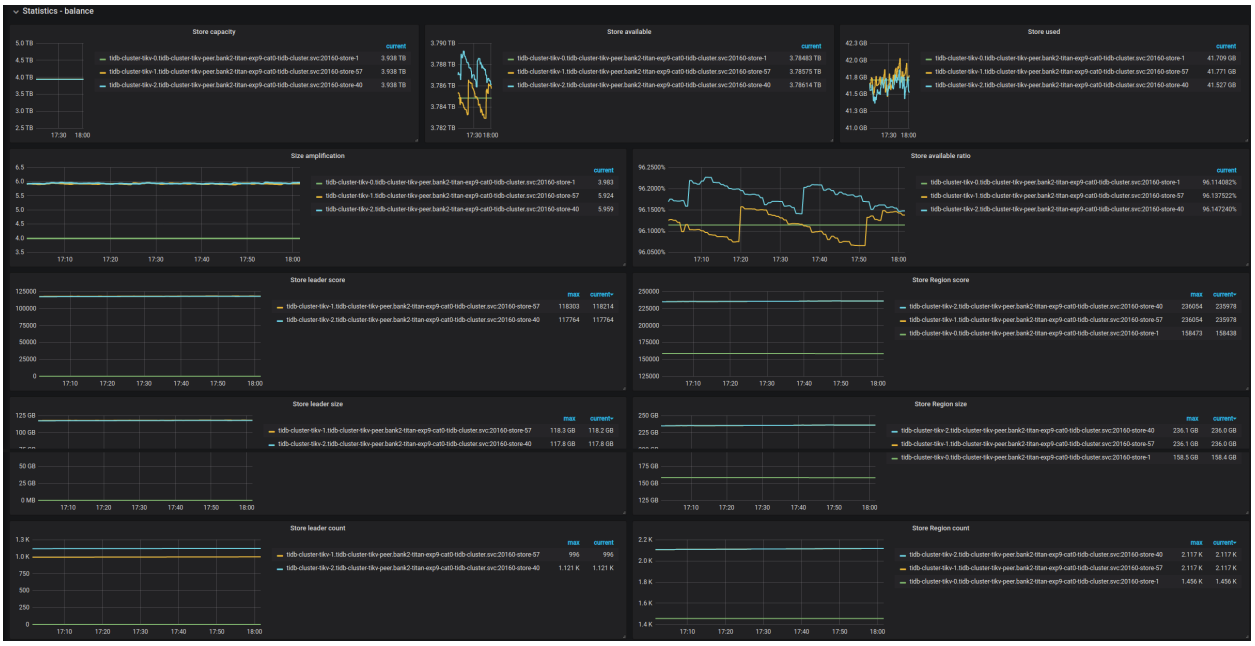

Figure 270: PD Dashboard - Statistics - Balance metrics

4.10.3.2.3 Statistics - Hotspot

Figure 271: PD Dashboard - Statistics - Hotspot metrics

4.10.3.2.4 Scheduler

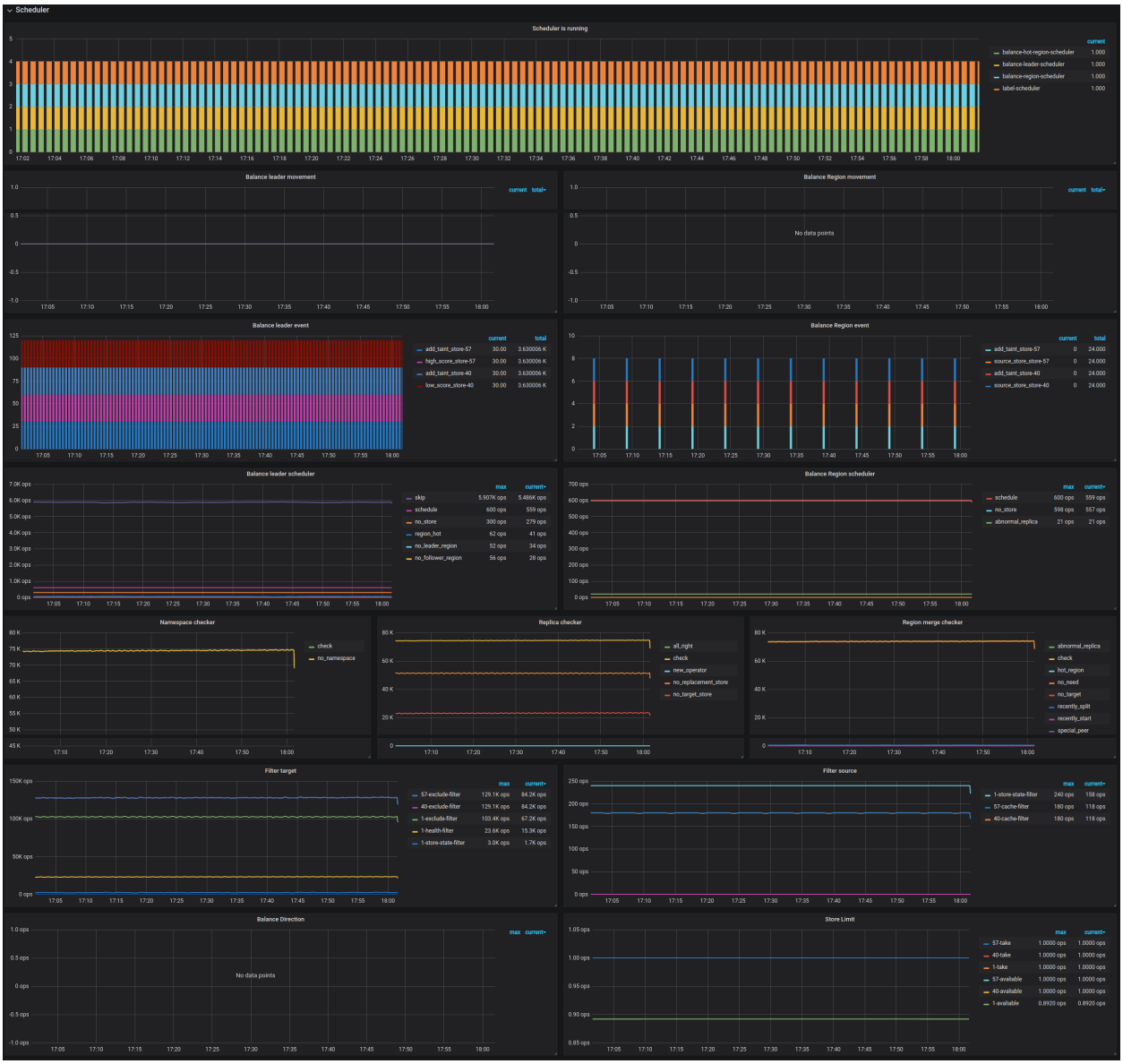

Figure 272: PD Dashboard - Scheduler metrics

4.10.3.2.5 Operator

Figure 273: PD Dashboard - Operator metrics

4.10.3.2.6 gRPC

Figure 274: PD Dashboard - gRPC metrics

4.10.3.2.7 etcd

Figure 275: PD Dashboard - etcd metrics

4.10.3.2.8 TiDB

Figure 276: PD Dashboard - TiDB metrics

4.10.3.2.9 Heartbeat

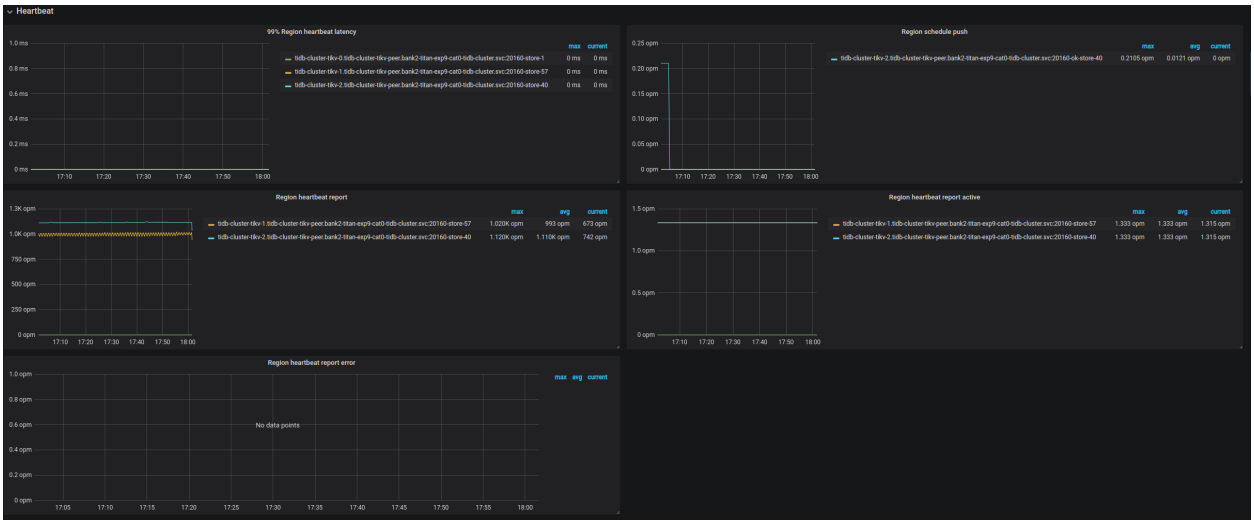

Figure 277: PD Dashboard - Heartbeat metrics

4.10.3.2.10 Region storage

Figure 278: PD Dashboard - Region storage

4.10.4 Key Monitoring Metrics of TiKV

If you use TiDB Ansible to deploy the TiDB cluster, the monitoring system is deployed at the same time. For more information, see Overview of the Monitoring Framework.

The Grafana dashboard is divided into a series of sub dashboards which include Overview, PD, TiDB, TiKV, Node_exporter, Disk Performance, and so on. A lot of metrics are there to help you diagnose.

You can get an overview of the component TiKV status from the TiKV dashboard, where the key metrics are displayed. This document provides a detailed description of these key metrics.

4.10.4.1 Key metrics description

To understand the key metrics displayed on the Overview dashboard, check the following table:

Service	Panel name	Description	Normal range
Cluster	Store size	The storage size per TiKV instance	
Cluster	Available size	The available capacity per TiKV instance	
Cluster	Capacity size	The capacity size per TiKV instance	
Cluster	CPU	The CPU usage per TiKV instance	
Cluster	Memory	The memory usage per TiKV instance	
Cluster	IO utilization	The I/O utilization per TiKV instance	
Cluster	MBps	The total bytes of read and write in each TiKV instance	
Cluster	QPS	The QPS per command in each TiKV instance	
Cluster	Errors-gRPC	The total number of gRPC message failures	
Cluster	Leaders	The number of leaders per TiKV instance	
Cluster	Regions	The number of Regions per TiKV instance	
Errors	Server is busy	Indicates occurrences of events that make the TiKV instance unavailable temporarily, such as Write Stall, Channel Full, Scheduler Busy, and Coprocessor Full	
Errors	Server message failures	The number of failed messages between TiKV instances	It should be `0` in normal case.
Errors	Raftstore errors	The number of Raftstore errors per type on each TiKV instance	
Errors	Scheduler errors	The number of scheduler errors per type on each TiKV instance	
Errors	Coprocessor errors	The number of coprocessor errors per type on each TiKV instance	
Errors	gRPC message errors	The number of gRPC message errors per type on each TiKV instance	
Errors	Leader drop	The count of dropped leaders per TiKV instance	

Service	Panel name	Description	Normal range
Errors	Leader missing	The count of missing leaders per TiKV instance	
Server	Leaders	The number of leaders per TiKV instance	
Server	Regions	The number of Regions per TiKV instance	
Server	CF size	The size of each column family	
Server	Store size	The storage size per TiKV instance	
Server	Channel full	The number of Channel Full errors per TiKV instance	It should be `0` in normal case.
Server	Server message failures	The number of failed messages between TiKV instances	
Server	Average Region written keys	The average rate of written keys to Regions per TiKV instance	
Server	Average Region written bytes	The average rate of writing bytes to Regions per TiKV instance	
Server	Active written leaders	The number of leaders being written on each TiKV instance	
Server	Approximate Region size	The approximate Region size	
Raft IO	Apply log duration	The time consumed for Raft to apply logs	
Raft IO	Apply log duration per server	The time consumed for Raft to apply logs per TiKV instance	
Raft IO	Append log duration	The time consumed for Raft to append logs	
Raft IO	Append log duration per server	The time consumed for Raft to append logs per TiKV instance	
Raft process	Ready handled	The count of handled ready buckets per region	
Raft process	Process ready duration per server	The time consumed for peer processes to be ready in Raft	It should be less than `2s` (P99.99).
Raft process	Process tick duration per server	The peer processes in Raft	
Raft process	99% Duration of raftstore events	The time consumed by raftstore events (P99)	

Service	Panel name	Description	Normal range
Raft message	Sent messages per server	The number of Raft messages sent by each TiKV instance	
Raft message	Flush messages per server	The number of Raft messages flushed by each TiKV instance	
Raft message	Receive messages per server	The number of Raft messages received by each TiKV instance	
Raft message	Messages	The number of Raft messages sent per type	
Raft message	Vote	The number of Vote messages sent in Raft	
Raft message	Raft dropped messages	The number of dropped Raft messages per type	
Raft proposal	Raft proposals per ready	The number of Raft proposals of all Regions per ready handled bucket	
Raft proposal	Raft read/write proposals	The number of proposals per type	
Raft proposal	Raft read proposals per server	The number of read proposals made by each TiKV instance	
Raft proposal	Raft write proposals per server	The number of write proposals made by each TiKV instance	
Raft proposal	Proposal wait duration	The wait time of each proposal	
Raft proposal	Proposal wait duration per server	The wait time of each proposal per TiKV instance	
Raft proposal	Raft log speed	The rate at which peers propose logs	
Raft admin	Admin proposals	The number of admin proposals	
Raft admin	Admin apply	The number of processed apply commands	
Raft admin	Check split	The number of raftstore split checks	
Raft admin	99.99% Check split duration	The time consumed when running split checks (P99.99)	
Local reader	Local reader requests	The number of total requests and the number of rejections from the local read thread	

Service	Panel name	Description	Normal range
Local reader	Local read requests duration	The wait time of local read requests	
Local reader	Local read requests batch size	The batch size of local read requests	
Storage	Storage command total	The total number of received commands per type	
Storage	Storage async request error	The total number of engine asynchronous request errors	
Storage	Storage async snapshot duration	The time consumed by processing asynchronous snapshot requests	It should be less than `1s` in `.99`.
Storage	Storage async write duration	The time consumed by processing asynchronous write requests	It should be less than `1s` in `.99`.
Scheduler	Scheduler stage total	The total number of commands at each stage	There should not be lots of errors in a short time.
Scheduler	Scheduler priority commands	The count of different priority commands	
Scheduler	Scheduler pending commands	The count of pending commands per TiKV instance	
Scheduler - XX	Scheduler stage total	The total number of commands at each stage when executing the batch_get command	There should not be lots of errors in a short time.
Scheduler - XX	Scheduler command duration	The time consumed when executing the batch_get command	It should be less than `1s`.
Scheduler - XX	Scheduler latch wait duration	The wait time caused by latch when executing the batch_get command	It should be less than `1s`.
Scheduler - XX	Scheduler keys read	The count of keys read by a batch_get command	
Scheduler - XX	Scheduler keys written	The count of keys written by a batch_get command	
Scheduler - XX	Scheduler scan details	The keys scan details of each CF when executing the batch_get command	

Service	Panel name	Description	Normal range
Scheduler - XX	Scheduler scan details [lock]	The keys scan details of lock CF when executing the batch_get command	
Scheduler - XX	Scheduler scan details [write]	The keys scan details of write CF when executing the batch_get command	
Scheduler - XX	Scheduler scan details [default]	The keys scan details of default CF when executing the batch_get command	
Coprocessor	Request duration	The time consumed to handle coprocessor read requests	
Coprocessor	Wait duration	The time consumed when coprocessor requests are waiting to be handled	It should be less than `10s` (P99.99).
Coprocessor	Processing duration	The time consumed to handle coprocessor requests	
Coprocessor	95% Request duration by store	The time consumed to handle coprocessor read requests per TiKV instance (P95)	
Coprocessor	95% Wait duration by store	The time consumed when coprocessor requests are waiting to be handled per TiKV instance (P95)	
Coprocessor	95% Handling duration by store	The time consumed to handle coprocessor requests per TiKV instance (P95)	
Coprocessor	Request errors	The total number of the push down request errors	There should not be lots of errors in a short time.
Coprocessor	DAG executors	The total number of DAG executors	
Coprocessor	Scan keys	The number of keys that each request scans	
Coprocessor	Scan details	The scan details for each CF	
Coprocessor	Table Scan - Details by CF	The table scan details for each CF	
Coprocessor	Index Scan - Details by CF	The index scan details for each CF	
Coprocessor	Table Scan - Perf Statistics	The total number of RocksDB internal operations from PerfContext when executing table scan	

Service	Panel name	Description	Normal range
Coprocessor	Index Scan - Perf Statistics	The total number of RocksDB internal operations from PerfContext when executing index scan	
GC	MVCC versions	The number of versions for each key	
GC	MVCC deleted versions	The number of versions deleted by GC for each key	
GC	GC tasks	The count of GC tasks processed by gc_worker	
GC	GC tasks Duration	The time consumed when executing GC tasks	
GC	GC keys (write CF)	The count of keys in write CF affected during GC	
GC	TiDB GC actions result	The TiDB GC action result on Region level	
GC	TiDB GC worker actions	The count of TiDB GC worker actions	
GC	TiDB GC seconds	The GC duration	
GC	TiDB GC failure	The count of failed TiDB GC jobs	
GC	GC lifetime	The lifetime of TiDB GC	
GC	GC interval	The interval of TiDB GC	
Snapshot	Rate snapshot message	The rate at which Raft snapshot messages are sent	
Snapshot	99% Handle snapshot duration	The time consumed to handle snapshots (P99)	
Snapshot	Snapshot state count	The number of snapshots per state	
Snapshot	99.99% Snapshot size	The snapshot size (P99.99)	
Snapshot	99.99% Snapshot KV count	The number of KV within a snapshot (P99.99)	
Task	Worker handled tasks	The number of tasks handled by worker	
Task	Worker pending tasks	Current number of pending and running tasks of worker	It should be less than 1000.
Task	FuturePool handled tasks	The number of tasks handled by future_pool	

Service	Panel name	Description	Normal range
Task	FuturePool pending tasks	Current number of pending and running tasks of future_pool	
Thread CPU	Raft store CPU	The CPU utilization of the raftstore thread	The CPU usage should be less than 80%.
Thread CPU	Async apply CPU	The CPU utilization of async apply	The CPU usage should be less than 90%.
Thread CPU	Scheduler CPU	The CPU utilization of scheduler	The CPU usage should be less than 80%.
Thread CPU	Scheduler Worker CPU	The CPU utilization of scheduler worker	
Thread CPU	Storage ReadPool CPU	The CPU utilization of readpool	
Thread CPU	Coprocessor CPU	The CPU utilization of coprocessor	
Thread CPU	Snapshot worker CPU	The CPU utilization of snapshot worker	
Thread CPU	Split check CPU	The CPU utilization of split check	
Thread CPU	RocksDB CPU	The CPU utilization of RocksDB	
Thread CPU	gRPC poll CPU	The CPU utilization of gRPC	The CPU usage should be less than 80%.
RocksDB - XX	Get operations	The count of get operations	
RocksDB - XX	Get duration	The time consumed when executing get operations	
RocksDB - XX	Seek operations	The count of seek operations	
RocksDB - XX	Seek duration	The time consumed when executing seek operations	
RocksDB - XX	Write operations	The count of write operations	
RocksDB - XX	Write duration	The time consumed when executing write operations	
RocksDB - XX	WAL sync operations	The count of WAL sync operations	
RocksDB - XX	WAL sync duration	The time consumed when executing WAL sync operations	

Service	Panel name	Description	Normal range
RocksDB - XX	Compaction operations	The count of compaction and flush operations	
RocksDB - XX	Compaction duration	The time consumed when executing the compaction and flush operations	
RocksDB - XX	SST read duration	The time consumed when reading SST files	
RocksDB - XX	Write stall duration	Write stall duration	It should be `0` in normal case.
RocksDB - XX	Memtable size	The memtable size of each column family	
RocksDB - XX	Memtable hit	The hit rate of memtable	
RocksDB - XX	Block cache size	The block cache size. Broken down by column family if shared block cache is disabled.	
RocksDB - XX	Block cache hit	The hit rate of block cache	
RocksDB - XX	Block cache flow	The flow rate of block cache operations per type	
RocksDB - XX	Block cache operations	The count of block cache operations per type	
RocksDB - XX	Keys flow	The flow rate of operations on keys per type	
RocksDB - XX	Total keys	The count of keys in each column family	
RocksDB - XX	Read flow	The flow rate of read operations per type	
RocksDB - XX	Bytes / Read	The bytes per read operation	
RocksDB - XX	Write flow	The flow rate of write operations per type	
RocksDB - XX	Bytes / Write	The bytes per write operation	
RocksDB - XX	Compaction flow	The flow rate of compaction operations per type	
RocksDB - XX	Compaction pending bytes	The pending bytes to be compacted	
RocksDB - XX	Read amplification	The read amplification per TiKV instance	
RocksDB - XX	Compression ratio	The compression ratio of each level	
RocksDB - XX	Number of snapshots	The number of snapshots per TiKV instance	

Service	Panel name	Description	Normal range
RocksDB - XX	Oldest snapshots duration	The time that the oldest unreleased snapshot survivals	
RocksDB - XX	Number files at each level	The number of SST files for different column families in each level	
RocksDB - XX	Ingest SST duration seconds	The time consumed to ingest SST files	
RocksDB - XX	Stall conditions changed of each CF	Stall conditions changed of each column family	
gRPC	gRPC messages	The count of gRPC messages per type	
gRPC	gRPC message failed	The count of failed gRPC messages per type	
gRPC	99% gRPC message duration	The gRPC message duration per message type (P99)	
gRPC	gRPC GC message count	The count of gRPC GC messages	
gRPC	99% gRPC KV GC message duration	The execution time of gRPC GC messages (P99)	
PD	PD requests	The count of requests that TiKV sends to PD	
PD	PD request duration (average)	The time consumed by requests that TiKV sends to PD	
PD	PD heartbeats	The total number of PD heartbeat messages	
PD	PD validated peers	The total number of peers validated by the PD worker	

4.10.4.2 TiKV dashboard interface

This section shows images of the service panels on the TiKV dashboard.

4.10.4.2.1 Cluster

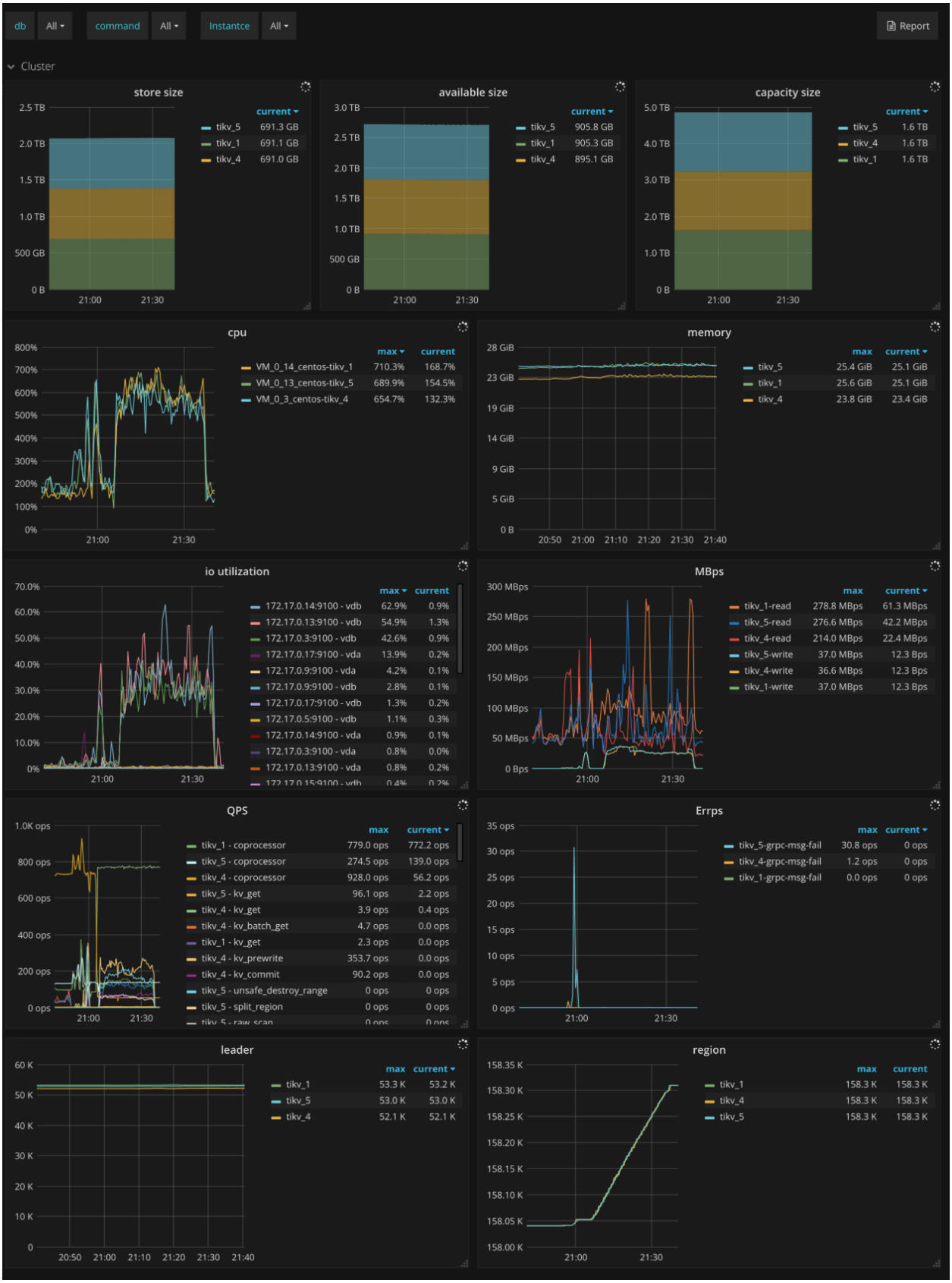

Figure 279: TiKV Dashboard - Cluster metrics

4.10.4.2.2 Errors

Figure 280: TiKV Dashboard - Errors metrics

4.10.4.2.3 Server

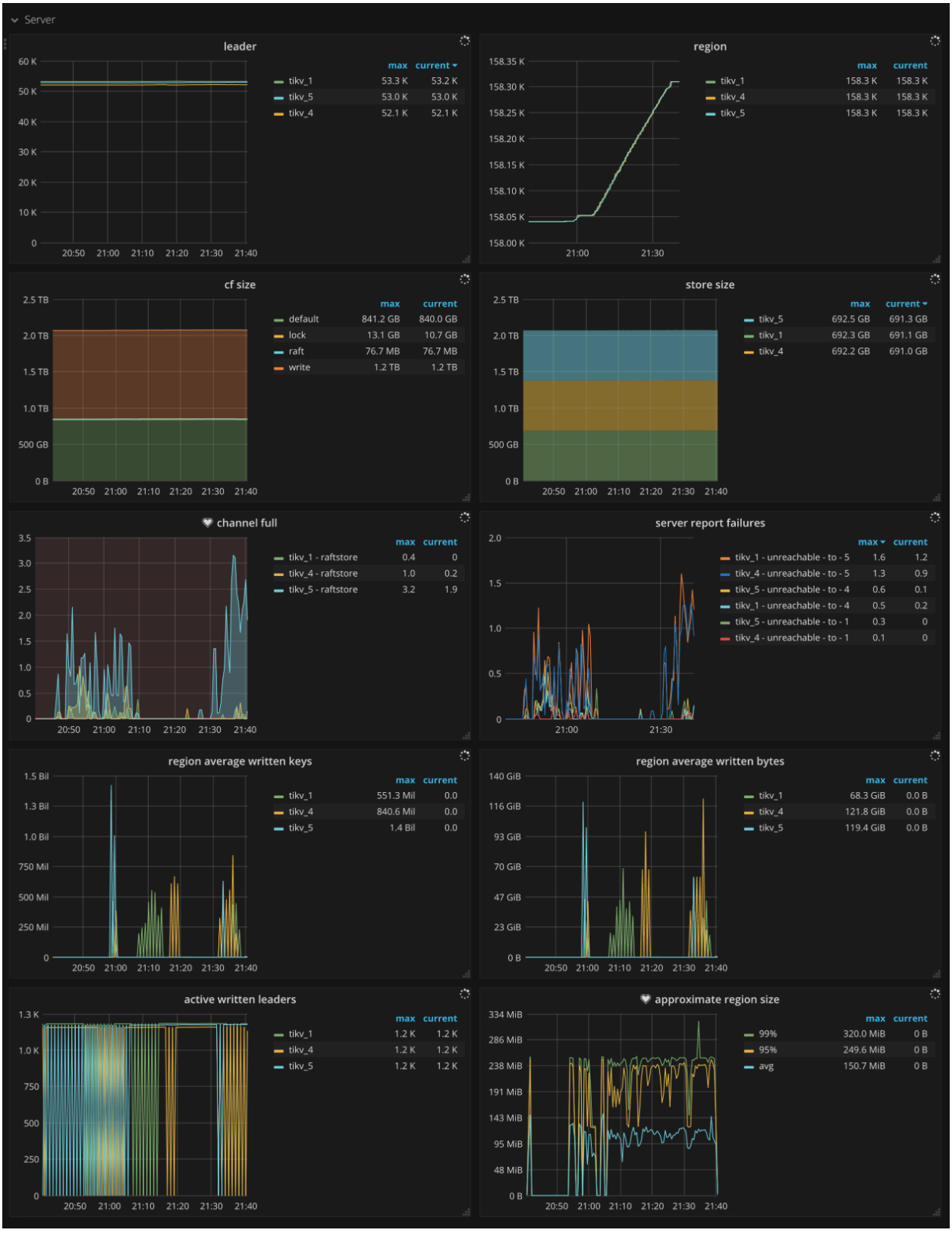

Figure 281: TiKV Dashboard - Server metrics

4.10.4.2.4 Raft IO

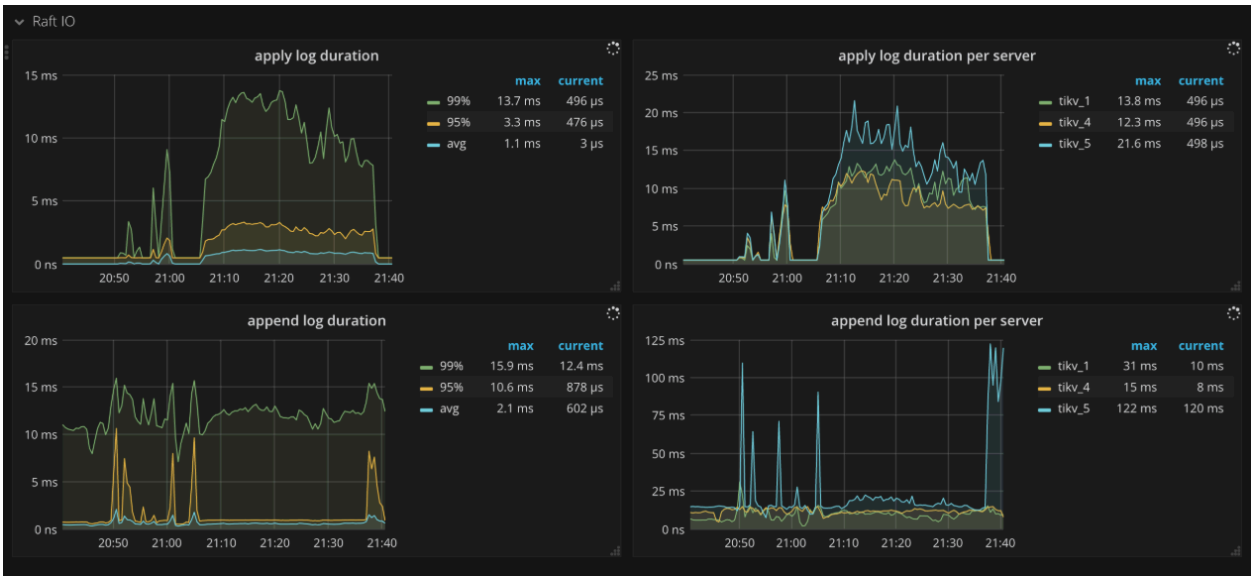

Figure 282: TiKV Dashboard - Raft IO metrics

4.10.4.2.5 Raft process

Figure 283: TiKV Dashboard - Raft process metrics

4.10.4.2.6 Raft message

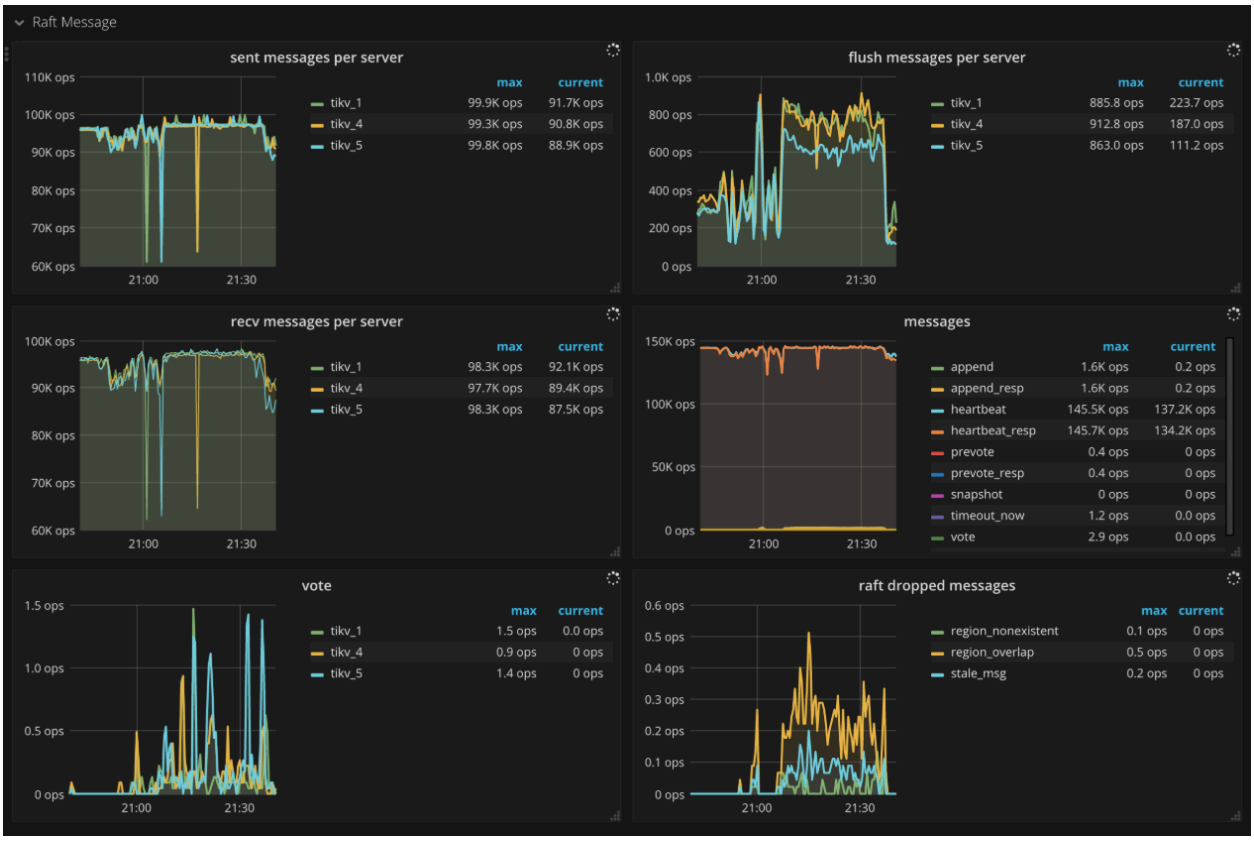

Figure 284: TiKV Dashboard - Raft message metrics

4.10.4.2.7 Raft proposal

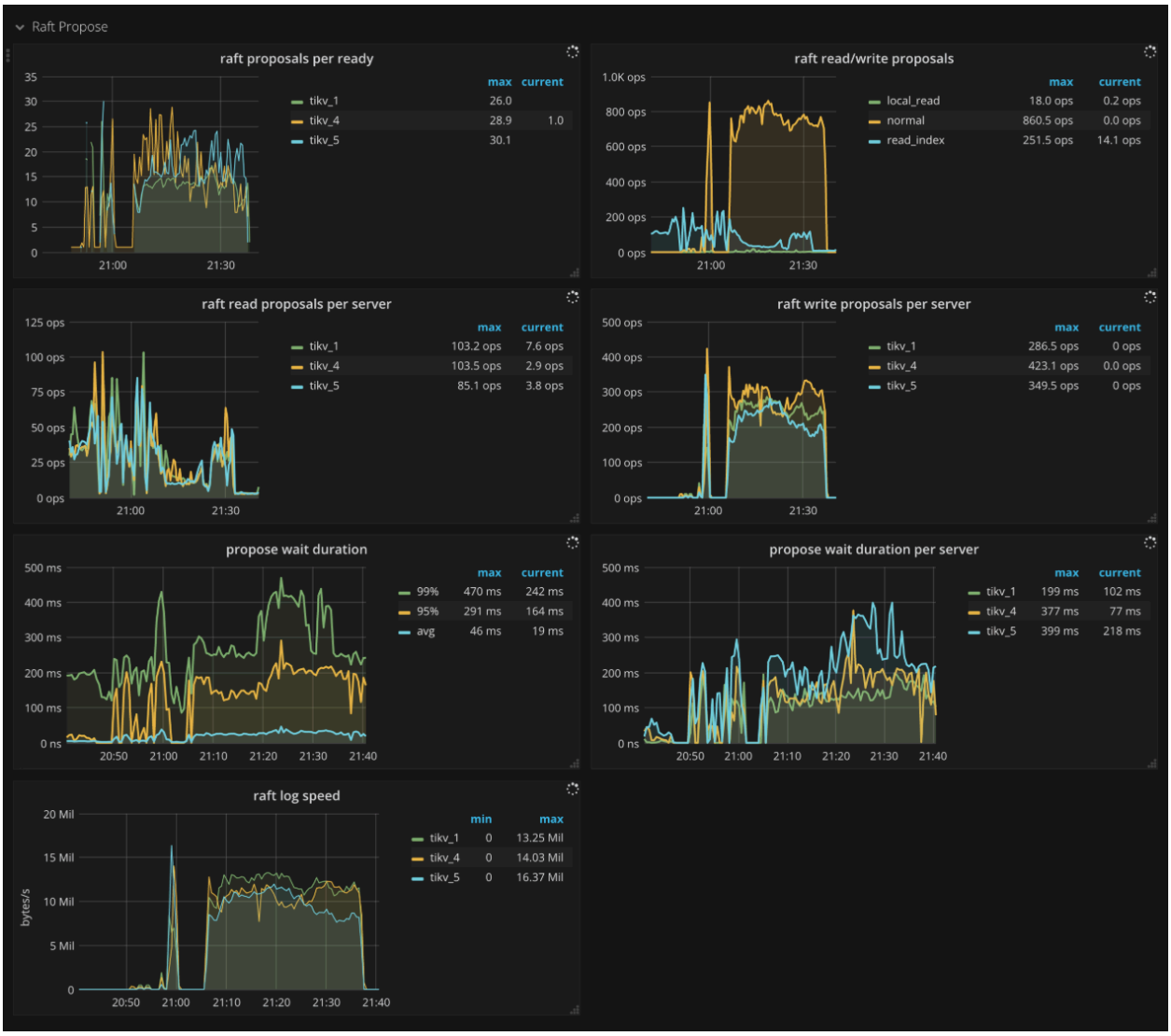

Figure 285: TiKV Dashboard - Raft proposal metrics

4.10.4.2.8 Raft admin

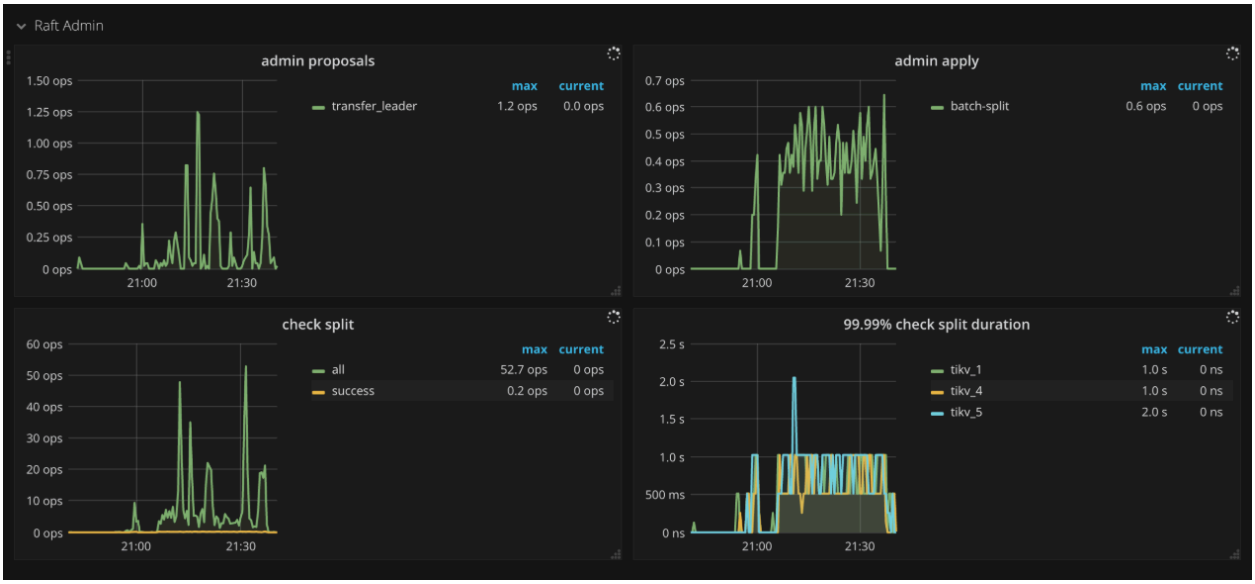

Figure 286: TiKV Dashboard - Raft admin metrics

4.10.4.2.9 Local reader

Figure 287: TiKV Dashboard - Local reader metrics

4.10.4.2.10 Storage

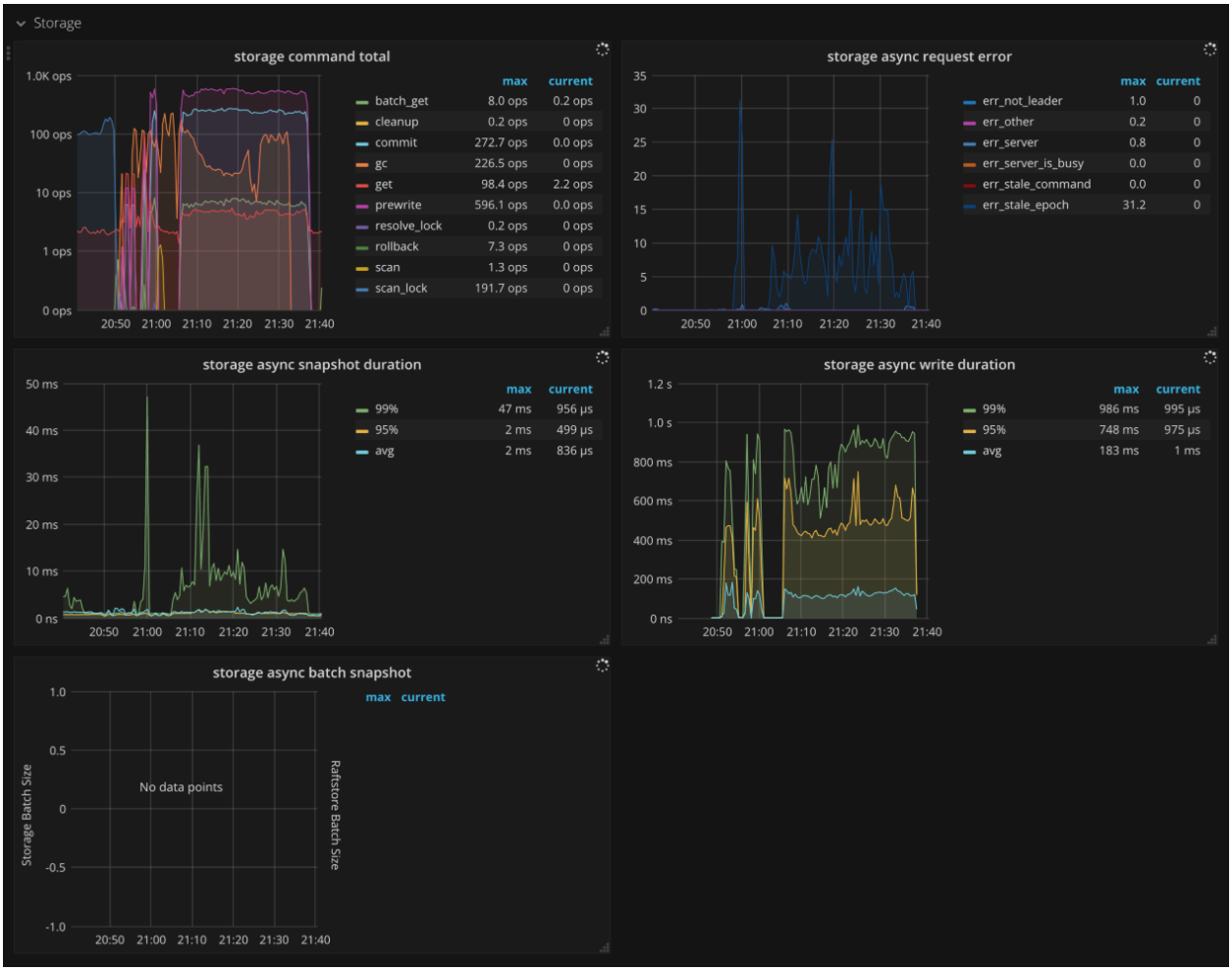

Figure 288: TiKV Dashboard - Storage metrics

4.10.4.2.11 Scheduler

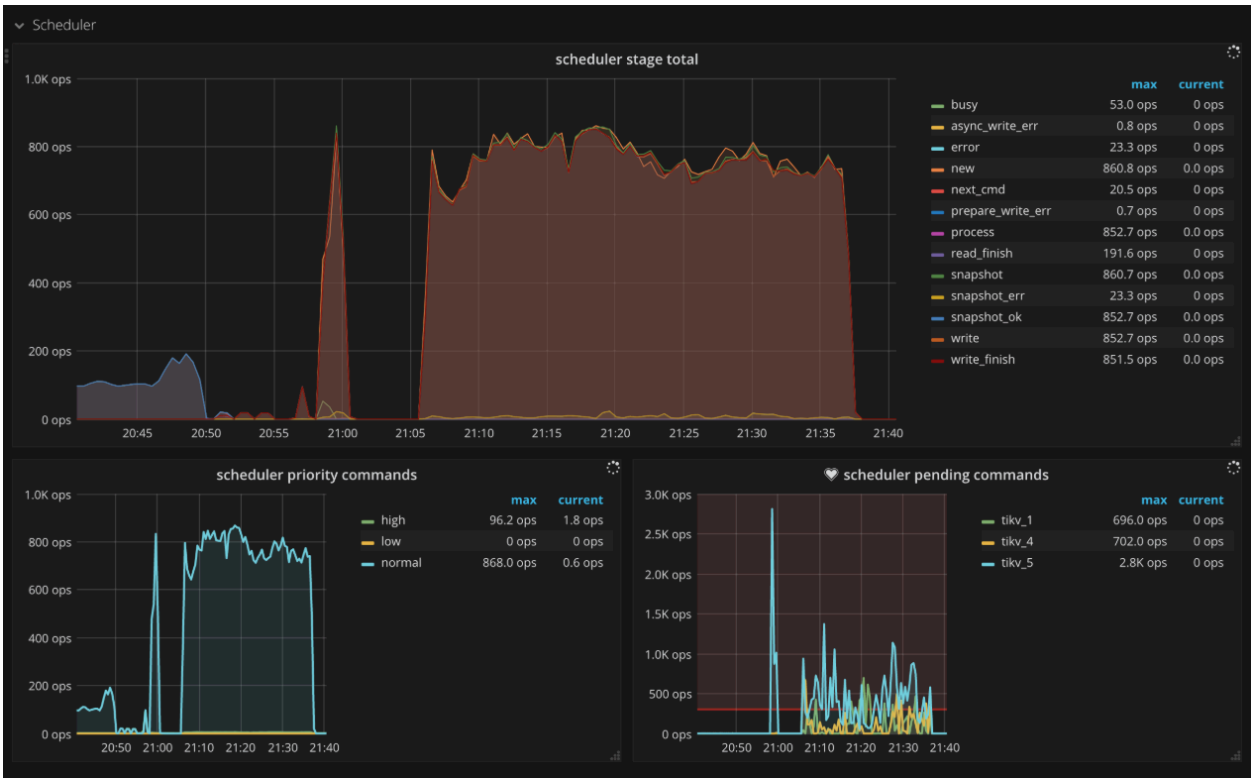

Figure 289: TiKV Dashboard - Scheduler metrics

4.10.4.2.12 Scheduler - batch_get

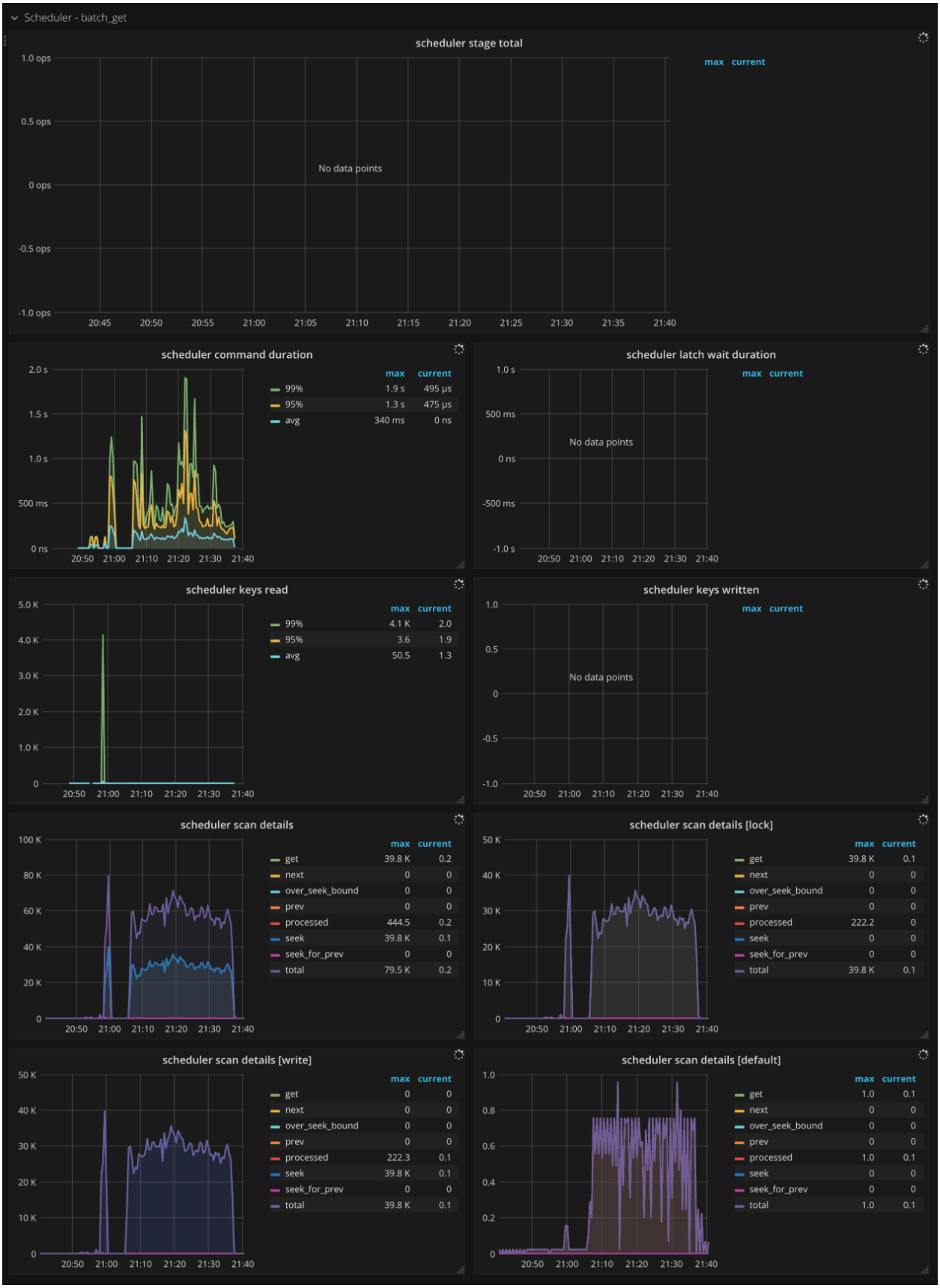

Figure 290: TiKV Dashboard - Scheduler - batch_get metrics

4.10.4.2.13 Scheduler - cleanup

Figure 291: TiKV Dashboard - Scheduler - cleanup metrics

4.10.4.2.14 Scheduler - commit

Figure 292: TiKV Dashboard - Scheduler commit metrics

4.11 TiDB Cluster Alert Rules

This document describes the alert rules for different components in a TiDB cluster, including the rule descriptions and solutions of the alert items in TiDB, TiKV, PD, TiDB Binlog, Node_exporter and Blackbox_exporter.

According to the severity level, alert rules are divided into three categories (from high to low): emergency-level, critical-level, and warning-level. This division of severity levels applies to all alert items of each component below.

Severity level	Description
Emergency-level	The highest severity level at which the service is unavailable. Emergency-level alerts are often caused by a service or node failure. **Manual intervention is required immediately**.

Severity level	Description
Critical-level	Decreased service availability. For the critical-level alerts, a close watch on the abnormal metrics is required.
Warning-level	Warning-level alerts are a reminder for an issue or error.

4.11.1 TiDB alert rules

This section gives the alert rules for the TiDB component.

4.11.1.1 Emergency-level alerts

4.11.1.1.1 `TiDB_schema_error`

- Alert rule:

```
increase(tidb_session_schema_lease_error_total{type="outdated"}[15m])>
↪ 0
```

- Description:

The latest schema information is not reloaded in TiDB within one lease. When TiDB fails to continue providing services, an alert is triggered.

- Solution:

  It is often caused by an unavailable Region or a TiKV timeout. You need to locate the issue by checking the TiKV monitoring items.

4.11.1.1.2 TiDB_tikvclient_region_err_total

- Alert rule:

  `increase(tidb_tikvclient_region_err_total[10m])> 6000`

- Description:

  When TiDB accesses TiKV, a Region error occurs. When the error is reported over 6000 times in 10 minutes, an alert is triggered.

- Solution:

  View the monitoring status of TiKV.

4.11.1.1.3 TiDB_domain_load_schema_total

- Alert rule:

  `increase(tidb_domain_load_schema_total{type="failed"}[10m])> 10`

- Description:

  The total number of failures to reload the latest schema information in TiDB. If the reloading failure occurs over 10 times in 10 minutes, an alert is triggered.

- Solution:

  Same as TiDB_schema_error.

4.11.1.1.4 TiDB_monitor_keep_alive

- Alert rule:

  `increase(tidb_monitor_keep_alive_total[10m])< 100`

- Description:

  Indicates whether the TiDB process still exists. If the number of times for `tidb_monitor_keep_alive_total` increases less than 100 in 10 minutes, the TiDB process might already exit and an alert is triggered.

- Solution:

  - Check whether the TiDB process is out of memory.
  - Check whether the machine has restarted.

4.11.1.2 Critical-level alerts

4.11.1.2.1 TiDB_server_panic_total

- Alert rule:

  `increase(tidb_server_panic_total[10m])> 0`

- Description:

  The number of panicked TiDB threads. When a panic occurs, an alert is triggered. The thread is often recovered, otherwise, TiDB will frequently restart.

- Solution:

  Collect the panic logs to locate the issue.

4.11.1.3 Warning-level alerts

4.11.1.3.1 TiDB_memory_abnormal

- Alert rule:

  `go_memstats_heap_inuse_bytes{job="tidb"} > 1e+10`

- Description:

  The monitoring on the TiDB memory usage. If the usage exceeds 10 G, an alert is triggered.

- Solution:

  Use the HTTP API to troubleshoot the goroutine leak issue.

4.11.1.3.2 TiDB_query_duration

- Alert rule:

  ```
  histogram_quantile(0.99, sum(rate(tidb_server_handle_query_duration_seconds_bucket
  ↪ [1m]))BY (le, instance))> 1
  ```

- Description:

  The latency of handling a request in TiDB. If the ninety-ninth percentile latency exceeds 1 second, an alert is triggered.

- Solution:

  View TiDB logs and search for the `SLOW_QUERY` and `TIME_COP_PROCESS` keywords to locate the slow SQL queries.

4.11.1.3.3 TiDB_server_event_error

- Alert rule:

  ```
  increase(tidb_server_event_total{type=~"server_start|server_hang"}[15m
  ↪ ])> 0
  ```

- Description:

  The number of events that happen in the TiDB service. An alert is triggered when the following events happen:

  1. start: The TiDB service starts.
  2. hang: When a critical-level event (currently there is only one scenario: TiDB cannot write binlog) happens, TiDB enters the `hang` mode and waits to be killed manually.

- Solution:

  - Restart TiDB to recover the service.
  - Check whether the TiDB Binlog service is normal.

4.11.1.3.4 TiDB_tikvclient_backoff_total

- Alert rule:

  ```
  increase(tidb_tikvclient_backoff_total[10m])> 10
  ```

- Description:

  The number of retries when TiDB fails to access TiKV. When the retry times is over 10 in 10 minutes, an alert is triggered.

- Solution:

  View the monitoring status of TiKV.

4.11.1.3.5 TiDB_monitor_time_jump_back_error

- Alert rule:

  ```
  increase(tidb_monitor_time_jump_back_total[10m])> 0
  ```

- Description:

  When the time of the machine that holds TiDB rewinds, an alert is triggered.

- Solution:

  Troubleshoot the NTP configurations.

4.11.1.3.6 `TiDB_ddl_waiting_jobs`

- Alert rule:

  `sum(tidb_ddl_waiting_jobs)> 5`

- Description:

  When the number of DDL tasks pending for execution in TiDB exceeds 5, an alert is triggered.

- Solution:

  Check whether there is any time-consuming `add index` operation that is being executed by running `admin show ddl`.

4.11.2 PD alert rules

This section gives the alert rules for the PD component.

4.11.2.1 Emergency-level alerts

4.11.2.1.1 `PD_cluster_offline_tikv_nums`

- Alert rule:

  `sum(pd_cluster_status{type="store_down_count"})> 0`

- Description:

  PD has not received a TiKV heartbeat for a long time (the default configuration is 30 minutes).

- Solution:

  - Check whether the TiKV process is normal, the network is isolated or the load is too high, and recover the service as much as possible.
  - If the TiKV instance cannot be recovered, you can make it offline.

4.11.2.2 Critical-level alerts

4.11.2.2.1 PD_etcd_write_disk_latency

- Alert rule:

```
histogram_quantile(0.99, sum(rate(etcd_disk_wal_fsync_duration_seconds_bucket
↪ [1m]))by (instance,job,le))> 1
```

- Description:

  etcd writes data to disk at a lower speed than normal. It might lead to PD leader timeout or failure to store TSO on disk in time, which will shut down the service of the entire cluster.

- Solution:

  - Find the cause of slow writes. It might be other services that overload the system. You can check whether PD itself occupies a large amount of CPU or I/O resources.
  - Try to restart PD or manually transfer leader to another PD to recover the service.
  - If the problematic PD instance cannot be recovered due to environmental factors, make it offline and replace it.

4.11.2.2.2 PD_miss_peer_region_count

- Alert rule:

```
sum(pd_regions_status{type="miss_peer_region_count"})> 100
```

- Description:

  The number of Region replicas is smaller than the value of `max-replicas`. When a TiKV machine is down and its downtime exceeds `max-down-time`, it usually leads to missing replicas for some Regions during a period of time. When a TiKV node is made offline, it might result in a small number of Regions with missing replicas.

- Solution:

  - Find the cause of the issue by checking whether there is any TiKV machine that is down or being made offline.
  - Watch the Region health panel and see whether `miss_peer_region_count` is continuously decreasing.

4.11.2.3 Warning-level alerts

4.11.2.3.1 PD_cluster_lost_connect_tikv_nums

- Alert rule:

  ```
  sum(pd_cluster_status{type="store_disconnected_count"})> 0
  ```

- Description:

  PD does not receive a TiKV heartbeat within 20 seconds. Normally a TiKV heartbeat comes in every 10 seconds.

- Solution:

  - Check whether the TiKV instance is being restarted.
  - Check whether the TiKV process is normal, the network is isolated, and the load is too high, and recover the service as much as possible.
  - If you confirm that the TiKV instance cannot be recovered, you can make it offline.
  - If you confirm that the TiKV instance can be recovered, but not in the short term, you can consider increasing the value of `max-down-time`. It will prevent the TiKV instance from being considered as irrecoverable and the data from being removed from the TiKV.

4.11.2.3.2 PD_cluster_low_space

- Alert rule:

  ```
  sum(pd_cluster_status{type="store_low_space_count"})> 0
  ```

- Description:

  Indicates that there is no sufficient space on the TiKV node.

- Solution:

  - Check whether the space in the cluster is generally insufficient. If so, increase its capacity.
  - Check whether there is any issue with Region balance scheduling. If so, it will lead to uneven data distribution.
  - Check whether there is any file that occupies a large amount of disk space, such as the log, snapshot, core dump, etc.
  - Lower the Region weight of the node to reduce the data volume.
  - When it is not possible to release the space, consider proactively making the node offline. This prevents insufficient disk space that leads to downtime.

4.11.2.3.3 PD_etcd_network_peer_latency

- Alert rule:

```
histogram_quantile(0.99, sum(rate(etcd_network_peer_round_trip_time_seconds_bucket
↪ [1m]))by (To,instance,job,le))> 1
```

- Description:

  The network latency between PD nodes is high. It might lead to the leader timeout and TSO disk storage timeout, which impacts the service of the cluster.

- Solution:

  - Check the network and system load status.
  - If the problematic PD instance cannot be recovered due to environmental factors, make it offline and replace it.

4.11.2.3.4 PD_tidb_handle_requests_duration

- Alert rule:

```
histogram_quantile(0.99, sum(rate(pd_client_request_handle_requests_duration_second
↪ {type="tso"}[1m]))by (instance,job,le))> 0.1
```

- Description:

  It takes a longer time for PD to handle the TSO request. It is often caused by a high load.

- Solution:

  - Check the load status of the server.
  - Use pprof to analyze the CPU profile of PD.
  - Manually switch the PD leader.
  - If the problematic PD instance cannot be recovered due to environmental factors, make it offline and replace it.

4.11.2.3.5 PD_down_peer_region_nums

- Alert rule:

```
sum(pd_regions_status{type="down_peer_region_count"})> 0
```

- Description:

  The number of Regions with an unresponsive peer reported by the Raft leader.

- Solution:

- Check whether there is any TiKV that is down, or that was just restarted, or that is busy.
- Watch the Region health panel and see whether `down_peer_region_count` is continuously decreasing.
- Check the network between TiKV servers.

4.11.2.3.6 `PD_pending_peer_region_count`

- Alert rule:

  ```
  sum(pd_regions_status{type="pending_peer_region_count"})> 100
  ```

- Description:

  There are too many Regions that have lagged Raft logs. It is normal that scheduling leads to a small number of pending peers, but if the number remains high, there might be an issue.

- Solution:

  - Watch the Region health panel and see whether `pending_peer_region_count` is continuously decreasing.
  - Check the network between TiKV servers, especially whether there is enough bandwidth.

4.11.2.3.7 `PD_leader_change`

- Alert rule:

  ```
  count(changes(pd_server_tso{type="save"}[10m])> 0)>= 2
  ```

- Description:

  The PD leader is recently switched.

- Solution:

  - Exclude the human factors, such as restarting PD, manually transferring leader, adjusting leader priority, etc.
  - Check the network and system load status.
  - If the problematic PD instance cannot be recovered due to environmental factors, make it offline and replace it.

4.11.2.3.8 TiKV_space_used_more_than_80%

- Alert rule:

```
sum(pd_cluster_status{type="storage_size"})/ sum(pd_cluster_status{type
↪ ="storage_capacity"})* 100 > 80
```

- Description:

  Over 80% of the cluster space is occupied.

- Solution:

  - Check whether it is needed to increase capacity.
  - Check whether there is any file that occupies a large amount of disk space, such as the log, snapshot, core dump, etc.

4.11.2.3.9 PD_system_time_slow

- Alert rule:

```
changes(pd_server_tso{type="system_time_slow"}[10m])>= 1
```

- Description:

  The system time rewind might happen.

- Solution:

  Check whether the system time is configured correctly.

4.11.2.3.10 PD_no_store_for_making_replica

- Alert rule:

```
increase(pd_checker_event_count{type="replica_checker", name="no_target_store
↪ "}[1m])> 0
```

- Description:

  There is no appropriate store for additional replicas.

- Solution:

  - Check whether there is enough space in the store.
  - Check whether there is any store for additional replicas according to the label configuration if it is configured.

4.11.3 TiKV alert rules

This section gives the alert rules for the TiKV component.

4.11.3.1 Emergency-level alerts

4.11.3.1.1 `TiKV_memory_used_too_fast`

- Alert rule:

```
process_resident_memory_bytes{job=~"tikv",instance=~".*"} - (process_resident_memor
↪ {job=~"tikv",instance=~".*"} offset 5m)> 5*1024*1024*1024
```

- Description:

  Currently, there are no TiKV monitoring items about memory. You can monitor the memory usage of the machines in the cluster by Node_exporter. The above rule indicates that when the memory usage exceeds 5 GB within 5 minutes (the memory is occupied too fast in TiKV), an alert is triggered.

- Solution:

  Adjust the `block-cache-size` value of both `rockdb.defaultcf` and `rocksdb.writecf` ↪ .

4.11.3.1.2 `TiKV_GC_can_not_work`

- Alert rule:

```
sum(increase(tikv_gcworker_gc_tasks_vec{task="gc"}[1d]))< 1
```

> **Note:**
> In TiDB 3.* versions, the `tidb_tikvclient_gc_action_result` metric exists but does not have a value. It's because distributed garbage collection (GC) is introduced in the TiDB 3.0 version but will not be performed in TiDB.

- Description:

  GC is not performed successfully on a TiKV instance within 24 hours, which indicates that GC is not working properly. If GC does not run in a short term, it will not cause much trouble; but if GC keeps down, more and more versions are retained, which slows down the query.

- Solution:

  1. Perform `select VARIABLE_VALUE from mysql.tidb where VARIABLE_NAME = ↪ "tikv_gc_leader_desc"` to locate the `tidb-server` corresponding to the GC leader;
  2. View the log of the `tidb-server`, and grep gc_worker tidb.log;

3. If you find that the GC worker has been resolving locks (the last log is "start resolve locks") or deleting ranges (the last log is "start delete {number} ranges") during this time, it means the GC process is running normally. Otherwise, contact support@pingcap.com to resolve this issue.

4.11.3.2 Critical-level alerts

4.11.3.2.1 `TiKV_server_report_failure_msg_total`

- Alert rule:

```
sum(rate(tikv_server_report_failure_msg_total{type="unreachable"}[10m])
↪ )BY (store_id)> 10
```

- Description:

  Indicates that the remote TiKV cannot be connected.

- Solution:

  1. Check whether the network is clear.
  2. Check whether the remote TiKV is down.
  3. If the remote TiKV is not down, check whether the pressure is too high. Refer to the solution in `TiKV_channel_full_total`.

4.11.3.2.2 `TiKV_channel_full_total`

- Alert rule:

```
sum(rate(tikv_channel_full_total[10m]))BY (type, instance)> 0
```

- Description:

  This issue is often caused by the stuck Raftstore thread and high pressure on TiKV.

- Solution:

  1. Watch the Raft Propose monitor, and see whether the alerted TiKV node has a much higher Raft propose than other TiKV nodes. If so, it means that there are one or more hot spots on this TiKV. You need to check whether the hot spot scheduling can work properly.
  2. Watch the Raft I/O monitor, and see whether the latency increases. If the latency is high, it means a bottleneck might exist in the disk. One feasible but unsafe solution is setting `sync-log` to `false`.
  3. Watch the Raft Process monitor, and see whether the tick duration is high. If so, you need to add `raft-base-tick-interval = "2s"` under the `[raftstore]` configuration.

4.11.3.2.3 TiKV_write_stall

- Alert rule:

  ```
  delta(tikv_engine_write_stall[10m])> 0
  ```

- Description:

  The write pressure on RocksDB is too high, and a stall occurs.

- Solution:

  1. View the disk monitor, and troubleshoot the disk issues;
  2. Check whether there is any write hot spot on the TiKV;
  3. Set `max-sub-compactions` to a larger value under the `[rocksdb]` and `[raftdb]` configurations.

4.11.3.2.4 TiKV_raft_log_lag

- Alert rule:

  ```
  histogram_quantile(0.99, sum(rate(tikv_raftstore_log_lag_bucket[1m]))by
  ↪  (le, instance))> 5000
  ```

- Description:

  If this value is relatively large, it means Follower has lagged far behind Leader, and Raft cannot be replicated normally. It is possibly because the TiKV machine where Follower is located is stuck or down.

4.11.3.2.5 TiKV_async_request_snapshot_duration_seconds

- Alert rule:

  ```
  histogram_quantile(0.99, sum(rate(tikv_storage_engine_async_request_duration_second
  ↪ {type="snapshot"}[1m]))by (le, instance, type))> 1
  ```

- Description:

  If this value is relatively large, it means the load pressure on Raftstore is too high, and it might be stuck already.

- Solution:

  Refer to the solution in TiKV_channel_full_total.

4.11.3.2.6 TiKV_async_request_write_duration_seconds

- Alert rule:

```
histogram_quantile(0.99, sum(rate(tikv_storage_engine_async_request_duration_second
↪ {type="write"}[1m]))by (le, instance, type))> 1
```

- Description:

  If this value is relatively large, it means the Raft write takes a long time.

- Solution:

  1. Check the pressure on Raftstore. See the solution in TiKV_channel_full_total.
  2. Check the pressure on the apply worker thread.

4.11.3.2.7 TiKV_coprocessor_request_wait_seconds

- Alert rule:

```
histogram_quantile(0.9999, sum(rate(tikv_coprocessor_request_wait_seconds_bucket
↪ [1m]))by (le, instance, req))> 10
```

- Description:

  If this value is relatively large, it means the pressure on the Coprocessor worker is high. There might be a slow task that makes the Coprocessor thread stuck.

- Solution:

  1. View the slow query log from the TiDB log to see whether the index or full table scan is used in a query, or see whether it is needed to analyze;
  2. Check whether there is a hot spot;
  3. View the Coprocessor monitor and see whether `total` and `process` in `coprocessor table/index scan` match. If they differ a lot, it indicates too many invalid queries are performed. You can see whether there is `over seek bound`. If so, there are too many versions that GC does not handle in time. Then you need to increase the number of parallel GC threads.

4.11.3.2.8 TiKV_raftstore_thread_cpu_seconds_total

- Alert rule:

```
sum(rate(tikv_thread_cpu_seconds_total{name=~"raftstore_.*"}[1m]))by (
↪ instance, name)> 1.6
```

- Description:

  The pressure on the Raftstore thread is too high.

- Solution:

  Refer to the solution in TiKV_channel_full_total.

4.11.3.2.9 TiKV_raft_append_log_duration_secs

- Alert rule:

```
histogram_quantile(0.99, sum(rate(tikv_raftstore_append_log_duration_seconds_bucket
↪ [1m]))by (le, instance))> 1
```

- Description:

  Indicates the time cost of appending Raft log. If it is high, it usually means I/O is too busy.

4.11.3.2.10 TiKV_raft_apply_log_duration_secs

- Alert rule:

```
histogram_quantile(0.99, sum(rate(tikv_raftstore_apply_log_duration_seconds_bucket
↪ [1m]))by (le, instance))> 1
```

- Description:

  Indicates the time cost of applying Raft log. If it is high, it usually means I/O is too busy.

4.11.3.2.11 TiKV_scheduler_latch_wait_duration_seconds

- Alert rule:

```
histogram_quantile(0.99, sum(rate(tikv_scheduler_latch_wait_duration_seconds_bucket
↪ [1m]))by (le, instance, type))> 1
```

- Description:

  The waiting time for the write operations to obtain the memory lock in Scheduler. If it is high, there might be many write conflicts, or that some operations that lead to conflicts take a long time to finish and block other operations that wait for the same lock.

- Solution:

  1. View the scheduler command duration in the Scheduler-All monitor and see which command is most time-consuming;
  2. View the scheduler scan details in the Scheduler-All monitor and see whether `total` and `process` match. If they differ a lot, there are many invalid scans. You can also see whether there is `over seek bound`. If there is too much, it indicates GC does not work in time;
  3. View the storage async snapshot/write duration in the Storage monitor and see whether the Raft operation is performed in time.

4.11.3.2.12 `TiKV_thread_apply_worker_cpu_seconds`

- Alert rule:

```
sum(rate(tikv_thread_cpu_seconds_total{name="apply_worker"}[1m]))by (
↪ instance)> 1.8
```

- Description:

  The pressure on the apply Raft log thread is too high. It is often caused by a burst of writes.

4.11.3.2.13 `TiDB_tikvclient_gc_action_fail` (only happen when in special configurations)

- Alert rule:

```
sum(increase(tidb_tikvclient_gc_action_result{type="fail" }[1m]))> 10
```

> **Note:**
> In TiDB 3.* versions, the `tidb_tikvclient_gc_action_result` metric exists but does not have a value. It's because distributed garbage collection (GC) is introduced in the TiDB 3.0 version but will not be performed in TiDB.

- Description:

  There are many Regions where GC fails to work.

- Solution:

  1. It is normally because the GC concurrency is set too high. You can moderately lower the GC concurrency degree, and you need to first confirm that the failed GC is caused by the busy server.
  2. You can moderately lower the concurrency degree by running `update set ↪ VARIABLE_VALUE="{number}" where VARIABLE_NAME=" tikv_gc_concurrency ↪ "` .

4.11.3.3 Warning-level alerts

4.11.3.3.1 TiKV_leader_drops

- Alert rule:

  ```
  delta(tikv_pd_heartbeat_tick_total{type="leader"}[30s])< -10
  ```

- Description:

  It is often caused by a stuck Raftstore thread.

- Solution:

  1. Refer to TiKV_channel_full_total.
  2. It there is low pressure on TiKV, consider whether the PD scheduling is too frequent. You can view the Operator Create panel on the PD page, and check the types and number of the PD scheduling.

4.11.3.3.2 TiKV_raft_process_ready_duration_secs

- Alert rule:

  ```
  histogram_quantile(0.999, sum(rate(tikv_raftstore_raft_process_duration_secs_bucket
  ↪ {type='ready'}[1m]))by (le, instance, type))> 2
  ```

- Description:

  Indicates the time cost of handling Raft ready. If this value is large, it is often caused by the stuck appending log task.

4.11.3.3.3 TiKV_raft_process_tick_duration_secs

- Alert rule:

  ```
  histogram_quantile(0.999, sum(rate(tikv_raftstore_raft_process_duration_secs_bucket
  ↪ {type='tick'}[1m]))by (le, instance, type))> 2
  ```

- Description:

  Indicates the time cost of handling Raft tick. If this value is large, it is often caused by too many Regions.

- Solution:

  1. Consider using a higher-level log such as `warn` or `error`.
  2. Add `raft-base-tick-interval = "2s"` under the `[raftstore]` configuration.

4.11.3.3.4 TiKV_scheduler_context_total

- Alert rule:

  ```
  abs(delta( tikv_scheduler_context_total[5m]))> 1000
  ```

- Description:

  The number of write commands that are being executed by Scheduler. If this value is large, it means the task is not finished timely.

- Solution:

  Refer to TiKV_scheduler_latch_wait_duration_seconds.

4.11.3.3.5 TiKV_scheduler_command_duration_seconds

- Alert rule:

  ```
  histogram_quantile(0.99, sum(rate(tikv_scheduler_command_duration_seconds_bucket
  ↪ [1m]))by (le, instance, type)/ 1000)> 1
  ```

- Description:

  Indicates the time cost of executing the Scheduler command.

- Solution:

  Refer to TiKV_scheduler_latch_wait_duration_seconds.

4.11.3.3.6 TiKV_coprocessor_outdated_request_wait_seconds

- Alert rule:

  ```
  delta(tikv_coprocessor_outdated_request_wait_seconds_count[10m])> 0
  ```

- Description:

  The waiting time of the expired requests by Coprocessor. If this value is large, it means there is high pressure on Coprocessor.

- Solution:

  Refer to TiKV_coprocessor_request_wait_seconds.

4.11.3.3.7 TiKV_coprocessor_request_error

- Alert rule:

  `increase(tikv_coprocessor_request_error{reason!="lock"}[10m])> 100`

- Description:

  The request error of Coprocessor.

- Solution:

  The reasons for the Coprocessor error can be divided into three types: "lock", "outdated" and "full". "outdated" indicates that the request has a timeout. It might be caused by a long queue time or a long time to handle a single request. "full" indicates that the request queue is full. It is possibly because the running request is time-consuming, which sends all new requests in the queue. You need to check whether the time-consuming query's execution plan is correct.

4.11.3.3.8 TiKV_coprocessor_request_lock_error

- Alert rule:

  `increase(tikv_coprocessor_request_error{reason="lock"}[10m])> 10000`

- Description:

  The lock requesting error of Coprocessor.

- Solution:

  The reasons for the Coprocessor error can be divided into three types: "lock", "outdated" and "full". "lock" indicates that the read data is being written and you need to wait a while and read again (the automatic retry happens inside TiDB). If just a few errors of this kind occur, you can ignore them; but if there are a lot of them, you need to check whether there is a conflict between the write and the query.

4.11.3.3.9 TiKV_coprocessor_pending_request

- Alert rule:

  `delta(tikv_coprocessor_pending_request[10m])> 5000`

- Description:

  The queuing requests of Coprocessor.

- Solution:

  Refer to TiKV_coprocessor_request_wait_seconds.

4.11.3.3.10 TiKV_batch_request_snapshot_nums

- Alert rule:

```
sum(rate(tikv_thread_cpu_seconds_total{name=~"cop_.*"}[1m]))by (instance
↪ )/ (count(tikv_thread_cpu_seconds_total{name=~"cop_.*"})* 0.9)/
↪ count(count(tikv_thread_cpu_seconds_total)by (instance))> 0
```

- Description:

  The Coprocessor CPU usage of a TiKV machine exceeds 90%.

4.11.3.3.11 TiKV_pending_task

- Alert rule:

```
sum(tikv_worker_pending_task_total)BY (instance,name)> 1000
```

- Description:

  The number of pending tasks of TiKV.

- Solution:

  Check which kind of tasks has a higher value. You can normally find a solution to the Coprocessor and apply worker tasks from other metrics.

4.11.3.3.12 TiKV_low_space_and_add_region

- Alert rule:

```
count((sum(tikv_store_size_bytes{type="available"})by (instance)/ sum(
↪ tikv_store_size_bytes{type="capacity"})by (instance)< 0.2)and (sum(
↪ tikv_raftstore_snapshot_traffic_total{type="applying"})by (instance)
↪ > 0))> 0
```

4.11.3.3.13 TiKV_approximate_region_size

- Alert rule:

```
histogram_quantile(0.99, sum(rate(tikv_raftstore_region_size_bucket[1m
↪ ]))by (le))> 1073741824
```

- Description:

  The maximum Region approximate size that is scanned by the TiKV split checker is continually larger than 1 GB within one minute.

- Solution:

  The speed of splitting Regions is slower than the write speed. To alleviate this issue, you'd better update TiDB to a version that supports batch-split (>= 2.1.0-rc1). If it is not possible to update temporarily, you can use `pd-ctl operator add split-region <region_id> --policy=approximate` to manually split Regions.

4.11.4 TiDB Binlog alert rules

For the detailed descriptions of TiDB Binlog alert rules, see TiDB Binlog monitoring document.

4.11.5 Node_exporter host alert rules

This section gives the alert rules for the Node_exporter host.

4.11.5.1 Emergency-level alerts

4.11.5.1.1 NODE_disk_used_more_than_80%

- Alert rule:

```
node_filesystem_avail_bytes{fstype=~"(ext.|xfs)", mountpoint!~"/boot"}
↪  / node_filesystem_size_bytes{fstype=~"(ext.|xfs)", mountpoint!~"/
↪ boot"} * 100 <= 20
```

- Description:

  The disk space usage of the machine exceeds 80%.

- Solution:

  - Log in to the machine, run the `df -h` command to check the disk space usage.
  - Make a plan to increase the disk capacity or delete some data or increase cluster node depending on different situations.

4.11.5.1.2 NODE_disk_inode_more_than_80%

- Alert rule:

```
node_filesystem_files_free{fstype=~"(ext.|xfs)"} / node_filesystem_files
↪ {fstype=~"(ext.|xfs)"} * 100 < 20
```

- Description:

  The inode usage of the filesystem on the machine exceeds 80%.

- Solution:

  - Log in to the machine and run the `df -i` command to view the inode usage of the filesystem.
  - Make a plan to increase the disk capacity or delete some data or increase cluster node depending on different situations.

4.11.5.1.3 NODE_disk_readonly

- Alert rule:

  ```
  node_filesystem_readonly{fstype=~"(ext.|xfs)"} == 1
  ```

- Description:

  The filesystem is read-only and data cannot be written in it. It is often caused by disk failure or filesystem corruption.

- Solution:

  - Log in to the machine and create a file to test whether it is normal.
  - Check whether the disk LED is normal. If not, replace the disk and repair the filesystem of the machine.

4.11.5.2 Critical-level alerts

4.11.5.2.1 NODE_memory_used_more_than_80%

- Alert rule:

  ```
  (((node_memory_MemTotal_bytes-node_memory_MemFree_bytes-node_memory_Cached_bytes
  ↪ )/(node_memory_MemTotal_bytes)*100))>= 80
  ```

- Description:

  The memory usage of the machine exceeds 80%.

- Solution:

  - View the Memory panel of the host in the Grafana Node Exporter dashboard, and see whether Used memory is too high and Available memory is too low.
  - Log in to the machine and run the `free -m` command to view the memory usage. You can run `top` to check whether there is any abnormal process that has an overly high memory usage.

4.11.5.3 Warning-level alerts

4.11.5.3.1 NODE_node_overload

- Alert rule:

  ```
  (node_load5 / count without (cpu, mode)(node_cpu_seconds_total{mode="
  ↪ system"}))> 1
  ```

- Description:

  The CPU load on the machine is relatively high.

- Solution:

  - View the CPU Usage and Load Average of the host in the Grafana Node Exporter dashboard to check whether they are too high.
  - Log in to the machine and run `top` to check the load average and the CPU usage, and see whether there is any abnormal process that has an overly high CPU usage.

4.11.5.3.2 NODE_cpu_used_more_than_80%

- Alert rule:

```
avg(irate(node_cpu_seconds_total{mode="idle"}[5m]))by(instance)* 100 <=
↪ 20
```

- Description:

  The CPU usage of the machine exceeds 80%.

- Solution:

  - View the CPU Usage and Load Average of the host on the Grafana Node Exporter dashboard to check whether they are too high.
  - Log in to the machine and run `top` to check the Load Average and the CPU Usage, and see whether there is any abnormal process that has an overly high CPU usage.

4.11.5.3.3 NODE_tcp_estab_num_more_than_50000

- Alert rule:

```
node_netstat_Tcp_CurrEstab > 50000
```

- Description:

  There are more than 50,000 TCP links in the "establish" status on the machine.

- Solution:

  - Log in to the machine and run `ss -s` to check the number of TCP links in the "estab" status in the current system.
  - Run `netstat` to check whether there is any abnormal link.

4.11.5.3.4 NODE_disk_read_latency_more_than_32ms

- Alert rule:

```
((rate(node_disk_read_time_seconds_total{device=~".+"}[5m])/ rate(
↪ node_disk_reads_completed_total{device=~".+"}[5m]))or (irate(node_disk_read_time
↪ {device=~".+"}[5m])/ irate(node_disk_reads_completed_total{device
↪ =~".+"}[5m])))* 1000 > 32
```

- Description:

  The read latency of the disk exceeds 32 ms.

- Solution:

  - Check the disk status by viewing the Grafana Disk Performance dashboard.
  - Check the read latency of the disk by viewing the Disk Latency panel.
  - Check the I/O usage by viewing the Disk I/O Utilization panel.

4.11.5.3.5 NODE_disk_write_latency_more_than_16ms

- Alert rule:

```
((rate(node_disk_write_time_seconds_total{device=~".+"}[5m])/ rate
↪ (node_disk_writes_completed_total{device=~".+"}[5m]))or (irate(
↪ node_disk_write_time_seconds_total{device=~".+"}[5m])/ irate(node_disk_writes_co
↪ {device=~".+"}[5m])))* 1000 > 16
```

- Description:

  The write latency of the disk exceeds 16ms.

- Solution:

  - Check the disk status by viewing the Grafana Disk Performance dashboard.
  - Check the write latency of the disk by viewing the Disk Latency panel.
  - Check the I/O usage by viewing the Disk I/O Utilization panel.

4.11.6 Blackbox_exporter TCP, ICMP, and HTTP alert rules

This section gives the alert rules for the Blackbox_exporter TCP, ICMP, and HTTP.

4.11.6.1 Emergency-level alerts

4.11.6.1.1 TiDB_server_is_down

- Alert rule:

  ```
  probe_success{group="tidb"} == 0
  ```

- Description:

  Failure to probe the TiDB service port.

- Solution:

  - Check whether the machine that provides the TiDB service is down.
  - Check whether the TiDB process exists.
  - Check whether the network between the monitoring machine and the TiDB machine is normal.

4.11.6.1.2 Pump_server_is_down

- Alert rule:

  ```
  probe_success{group="pump"} == 0
  ```

- Description:

  Failure to probe the pump service port.

- Solution:

  - Check whether the machine that provides the pump service is down.
  - Check whether the pump process exists.
  - Check whether the network between the monitoring machine and the pump machine is normal.

4.11.6.1.3 Drainer_server_is_down

- Alert rule:

  ```
  probe_success{group="drainer"} == 0
  ```

- Description:

  Failure to probe the Drainer service port.

- Solution:

  - Check whether the machine that provides the Drainer service is down.
  - Check whether the Drainer process exists.
  - Check whether the network between the monitoring machine and the Drainer machine is normal.

4.11.6.1.4 TiKV_server_is_down

- Alert rule:

  `probe_success{group="tikv"} == 0`

- Description:

  Failure to probe the TiKV service port.

- Solution:

  - Check whether the machine that provides the TiKV service is down.
  - Check whether the TiKV process exists.
  - Check whether the network between the monitoring machine and the TiKV machine is normal.

4.11.6.1.5 PD_server_is_down

- Alert rule:

  `probe_success{group="pd"} == 0`

- Description:

  Failure to probe the PD service port.

- Solution:

  - Check whether the machine that provides the PD service is down.
  - Check whether the PD process exists.
  - Check whether the network between the monitoring machine and the PD machine is normal.

4.11.6.1.6 Node_exporter_server_is_down

- Alert rule:

  `probe_success{group="node_exporter"} == 0`

- Description:

  Failure to probe the Node_exporter service port.

- Solution:

  - Check whether the machine that provides the Node_exporter service is down.
  - Check whether the Node_exporter process exists.
  - Check whether the network between the monitoring machine and the Node_exporter machine is normal.

4.11.6.1.7 Blackbox_exporter_server_is_down

- Alert rule:

  ```
  probe_success{group="blackbox_exporter"} == 0
  ```

- Description:

  Failure to probe the Blackbox_Exporter service port.

- Solution:

  - Check whether the machine that provides the Blackbox_Exporter service is down.
  - Check whether the Blackbox_Exporter process exists.
  - Check whether the network between the monitoring machine and the Blackbox_Exporter machine is normal.

4.11.6.1.8 Grafana_server_is_down

- Alert rule:

  ```
  probe_success{group="grafana"} == 0
  ```

- Description:

  Failure to probe the Grafana service port.

- Solution:

  - Check whether the machine that provides the Grafana service is down.
  - Check whether the Grafana process exists.
  - Check whether the network between the monitoring machine and the Grafana machine is normal.

4.11.6.1.9 Pushgateway_server_is_down

- Alert rule:

  ```
  probe_success{group="pushgateway"} == 0
  ```

- Description:

  Failure to probe the Pushgateway service port.

- Solution:

  - Check whether the machine that provides the Pushgateway service is down.
  - Check whether the Pushgateway process exists.
  - Check whether the network between the monitoring machine and the Pushgateway machine is normal.

4.11.6.1.10 Kafka_exporter_is_down

- Alert rule:

  ```
  probe_success{group="kafka_exporter"} == 0
  ```

- Description:

  Failure to probe the Kafka_Exporter service port.

- Solution:

  - Check whether the machine that provides the Kafka_Exporter service is down.
  - Check whether the Kafka_Exporter process exists.
  - Check whether the network between the monitoring machine and the Kafka_Exporter machine is normal.

4.11.6.1.11 Pushgateway_metrics_interface

- Alert rule:

  ```
  probe_success{job="blackbox_exporter_http"} == 0
  ```

- Description:

  Failure to probe the Pushgateway service http interface.

- Solution:

  - Check whether the machine that provides the Pushgateway service is down.
  - Check whether the Pushgateway process exists.
  - Check whether the network between the monitoring machine and the Pushgateway machine is normal.

4.11.6.2 Warning-level alerts

4.11.6.2.1 BLACKER_ping_latency_more_than_1s

- Alert rule:

  ```
  max_over_time(probe_duration_seconds{job=~"blackbox_exporter.*_icmp"}[1
  ↪ m])> 1
  ```

- Description:

  The ping latency exceeds 1 second.

- Solution:

  - View the ping latency between the two nodes on the Grafana Blackbox Exporter dashboard to check whether it is too high.
  - Check the tcp panel on the Grafana Node Exporter dashboard to check whether there is any packet loss.

4.12 Best Practices

4.12.1 Highly Concurrent Write Best Practices

This document describes best practices for handling highly-concurrent write-heavy workloads in TiDB, which can help to facilitate your application development.

4.12.1.1 Target audience

This document assumes that you have a basic understanding of TiDB. It is recommended that you first read the following three blog articles that explain TiDB fundamentals, and TiDB Best Practices:

- Data Storage
- Computing
- Scheduling

4.12.1.2 Highly-concurrent write-intensive scenario

The highly concurrent write scenario often occurs when you perform batch tasks in applications, such as clearing, settlement and so on. This scenario has the following features:

- A huge volume of data
- The need to import historical data into database in a short time
- The need to read a huge volume of data from database in a short time

These features pose these challenges to TiDB:

- The write or read capacity must be linearly scalable.
- Database performance is stable and does not decrease as a huge volume of data is written concurrently.

For a distributed database, it is important to make full use of the capacity of all nodes and to prevent a single node from becoming the bottleneck.

4.12.1.3 Data distribution principles in TiDB

To address the above challenges, it is necessary to start with the data segmentation and scheduling principle of TiDB. Refer to Scheduling for more details.

TiDB splits data into Regions, each representing a range of data with a size limit of 96M by default. Each Region has multiple replicas, and each group of replicas is called a Raft Group. In a Raft Group, the Region Leader executes the read and write tasks within the data range. The Region Leader is automatically scheduled by the Placement Driver (PD) component to different physical nodes evenly to distribute the read and write pressure.

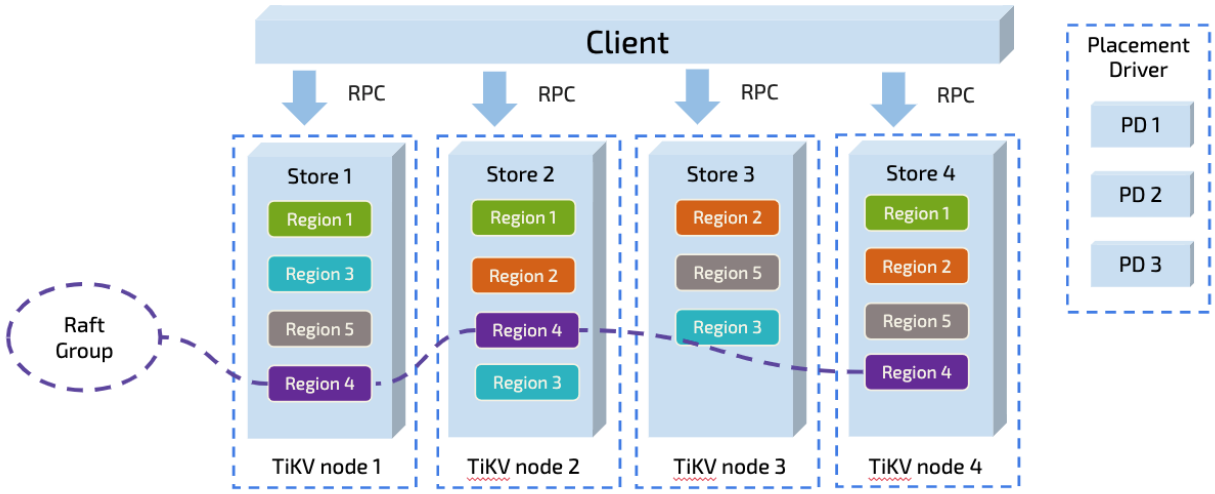

Figure 293: TiDB Data Overview

In theory, by the virtue of this architecture, TiDB can linearly scale its read and write capacities and make full use of the distributed resources so long as there is no `AUTO_INCREMENT` primary key in the write scenario, or there is no monotonically increasing index. From this point of view, TiDB is especially suitable for the highly-concurrent and write-intensive scenario. However, the actual situation often differs from the theoretical assumption.

> **Note:**
>
> No `AUTO_INCREMENT` primary key in the write scenario or no monotonically increasing index means no write hotspot in the application.

4.12.1.4 Hotspot case

The following case explains how a hotspot is generated. Take the table below as an example:

```
CREATE TABLE IF NOT EXISTS TEST_HOTSPOT(
    id        BIGINT PRIMARY KEY,
    age       INT,
    user_name VARCHAR(32),
    email     VARCHAR(128)
)
```

This table is simple in structure. In addition to `id` as the primary key, no secondary index exists. Execute the following statement to write data into this table. `id` is discretely generated as a random number.

```
INSERT INTO TEST_HOTSPOT(id, age, user_name, email) values(%v, %v, '%v', '%
    ↪ v');
```

The load comes from executing the above statement intensively in a short time.

In theory, the above operation seems to comply with the TiDB best practices, and no hotspot is caused in the application. The distributed capacity of TiDB can be fully used with adequate machines. To verify whether it is truly in line with the best practices, a test is conducted in the experimental environment, which is described as follows:

For the cluster topology, 2 TiDB nodes, 3 PD nodes and 6 TiKV nodes are deployed. Ignore the QPS performance, because this test is to clarify the principle rather than for benchmark.

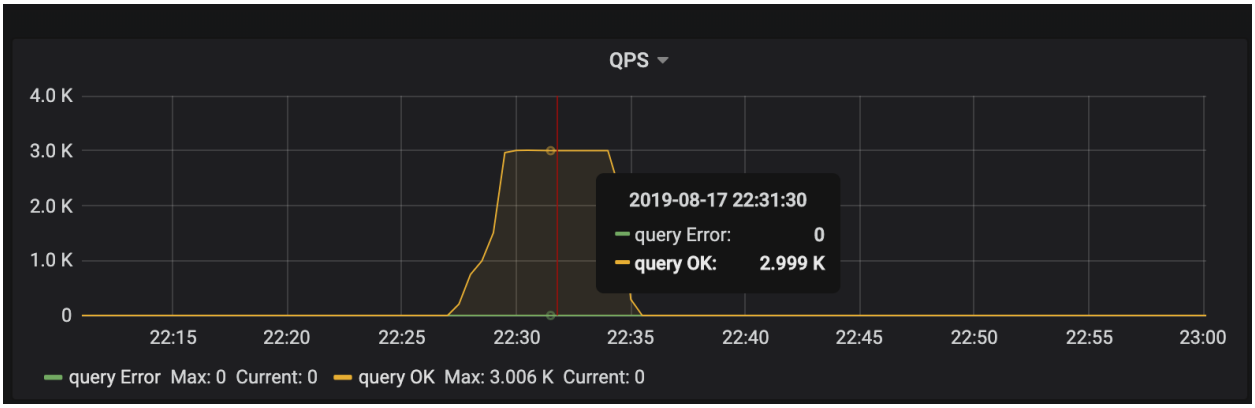

Figure 294: QPS1

The client starts "intensive" write requests in a short time, which is 3K QPS received by TiDB. In theory, the load pressure should be evenly distributed to 6 TiKV nodes. However, from the CPU usage of each TiKV node, the load distribution is uneven. The `tikv-3` node is the write hotspot.

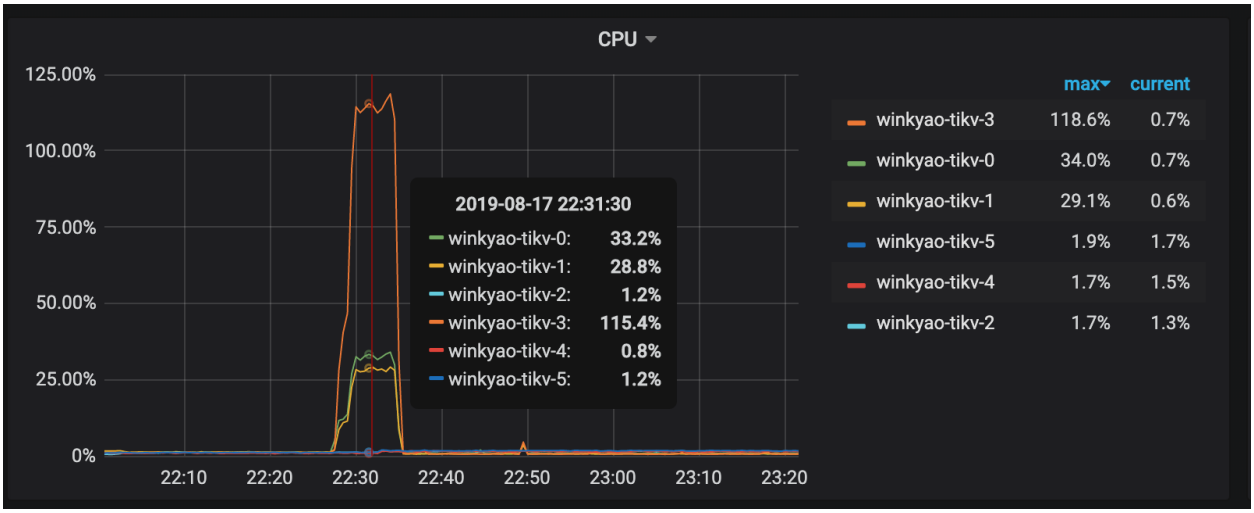

Figure 295: QPS2

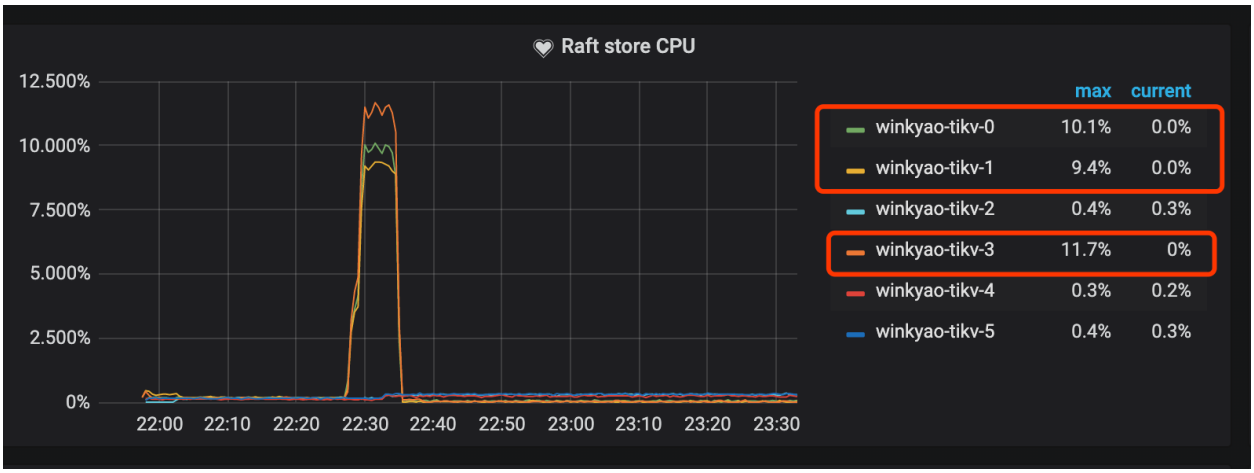

Figure 296: QPS3

Raft store CPU is the CPU usage rate for the `raftstore` thread, usually representing the write load. In this scenario, `tikv-3` is the Leader of this Raft Group; `tikv-0` and `tikv-1` are the followers. The loads of other nodes are almost empty.

The monitoring metrics of PD also confirms that hotspot has been caused.

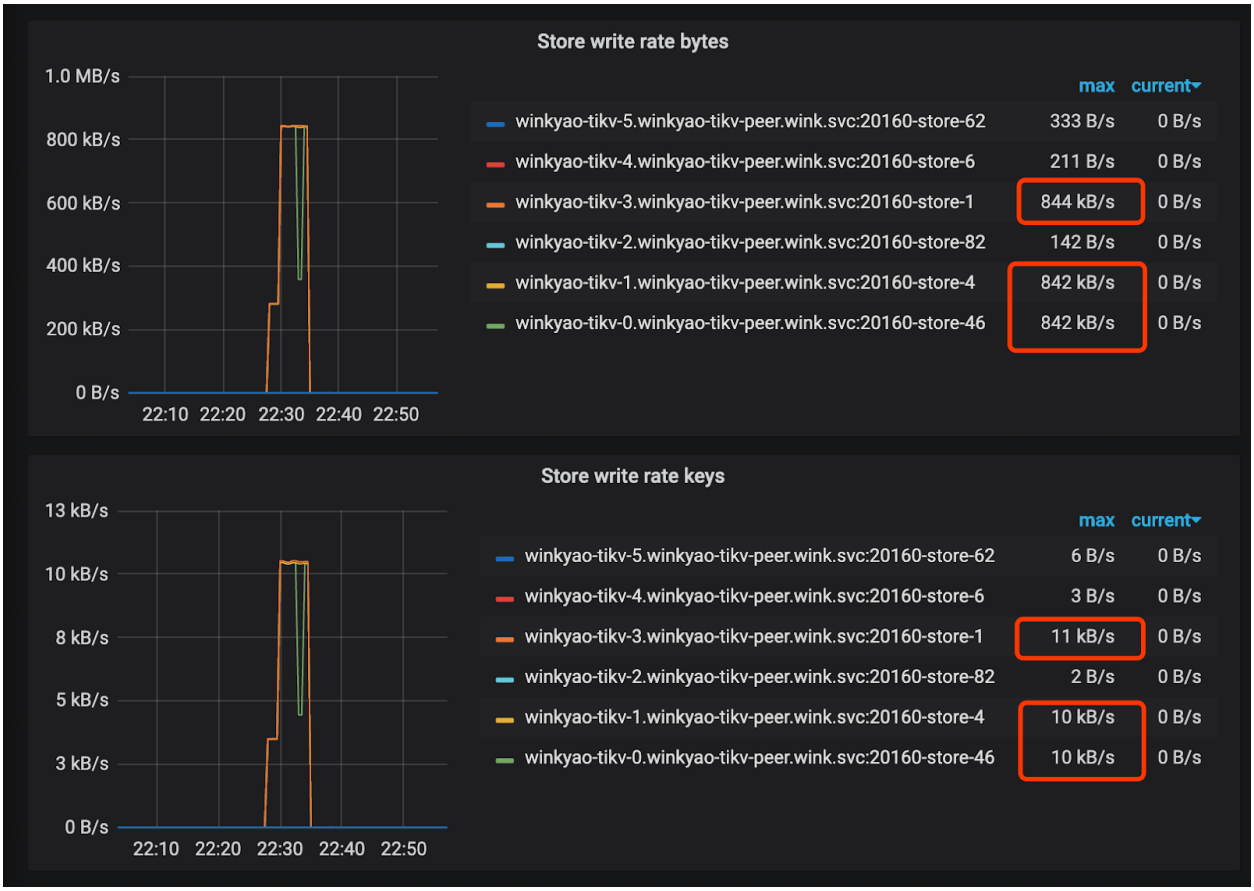

Figure 297: QPS4

4.12.1.5 Hotspot causes

In the above test, the operation does not reach the ideal performance expected in the best practices. This is because only one Region is split by default to store the data of each newly created table in TiDB, with the following data range:

```
[CommonPrefix + TableID, CommonPrefix + TableID + 1)
```

In a short period of time, a huge volume of data is continuously written to the same Region.

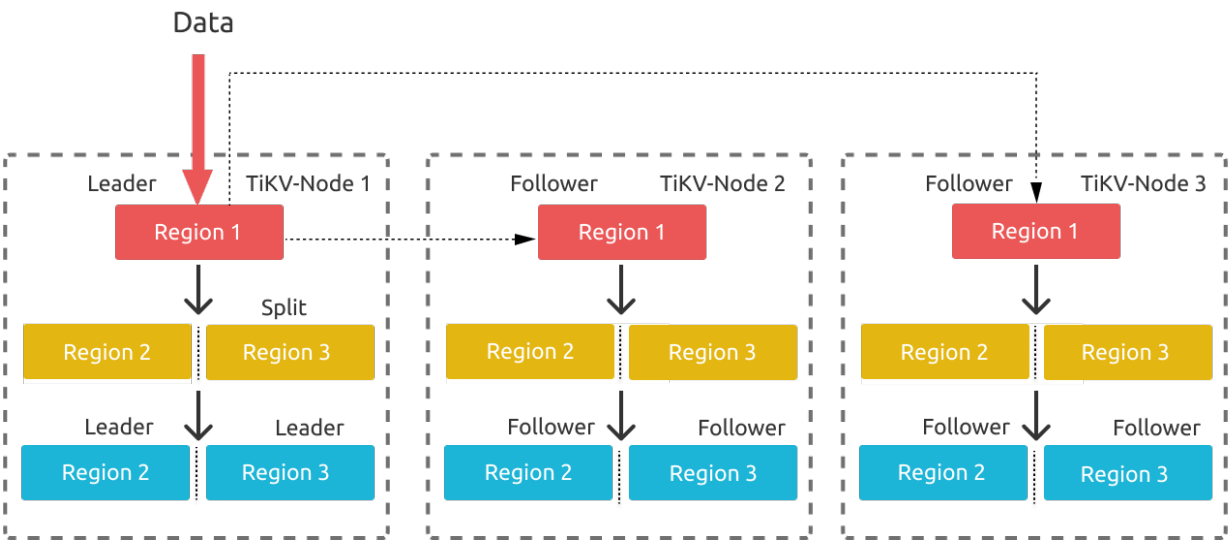

Figure 298: TiKV Region Split

The above diagram illustrates the Region splitting process. As data is continuously written into TiKV, TiKV splits a Region into multiple Regions. Because the leader election is started on the original store where the Region Leader to be split is located, the leaders of the two newly split Regions might be still on the same store. This splitting process might also happen on the newly split Region 2 and Region 3. In this way, write pressure is concentrated on TiKV-Node 1.

During the continuous write process, after finding that hotspot is caused on Node 1, PD evenly distributes the concentrated Leaders to other nodes. If the number of TiKV nodes is more than the number of Region replicas, TiKV will try to migrate these Regions to idle nodes. These two operations during the write process are also reflected in the PD's monitoring metrics:

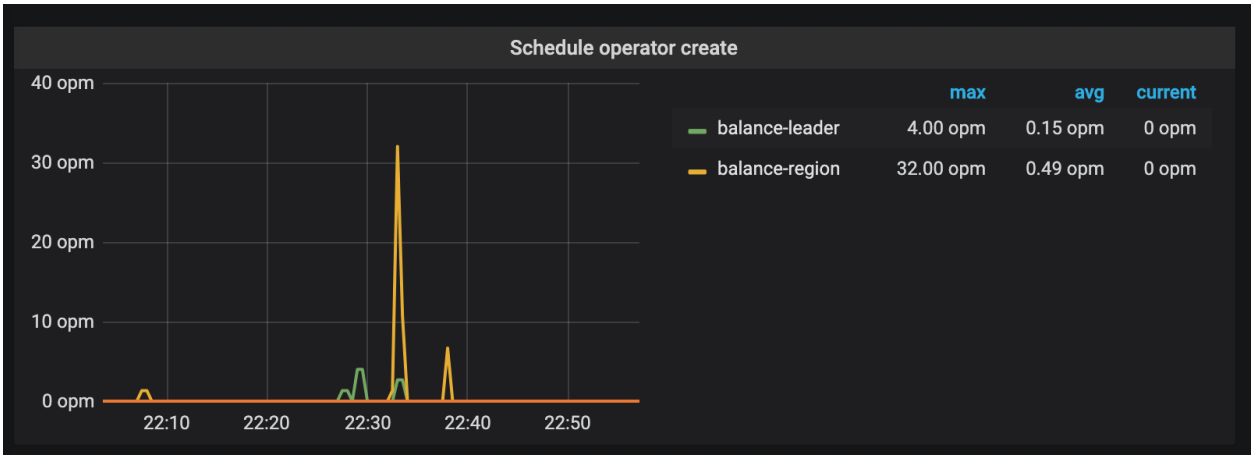

Figure 299: QPS5

After a period of continuous writes, PD automatically schedules the entire TiKV cluster to a state where pressure is evenly distributed. By that time, the capacity of the whole cluster can be fully used.

In most cases, the above process of causing a hotspot is normal, which is the Region warm-up phase of database. However, you need to avoid this phase in highly-concurrent write-intensive scenarios.

4.12.1.6 Hotspot solution

To achieve the ideal performance expected in theory, you can skip the warm-up phase by directly splitting a Region into the desired number of Regions and scheduling these Regions in advance to other nodes in the cluster.

In v3.0.x, v2.1.13 and later versions, TiDB supports a new feature called Split Region. This new feature provides the following new syntaxes:

```
SPLIT TABLE table_name [INDEX index_name] BETWEEN (lower_value) AND (
    ↪ upper_value) REGIONS region_num
```

```
SPLIT TABLE table_name [INDEX index_name] BY (value_list) [, (value_list)]
```

However, TiDB does not automatically perform this pre-split operation. The reason is related to the data distribution in TiDB.

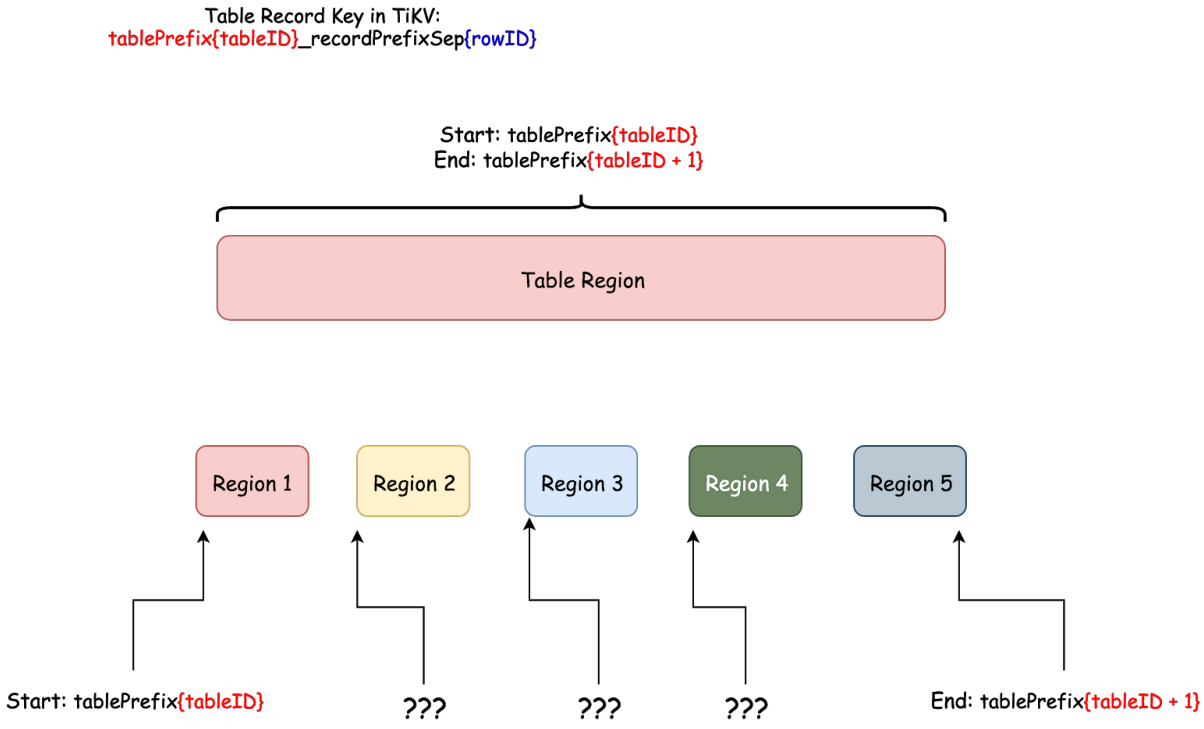

Figure 300: Table Region Range

From the diagram above, according to the encoding rule of a row's key, the `rowID` is the only variable part. In TiDB, `rowID` is an `Int64` integer. However, you might not need to evenly split the `Int64` integer range to the desired number of ranges and then to distribute these ranges to different nodes, because Region split must also be based on the actual situation.

If the write of `rowID` is completely discrete, the above method will not cause hotspots. If the row ID or index has a fixed range or prefix (for example, discretely insert data into the range of `[2000w, 5000w)`), no hotspot will be caused either. However, if you split a Region using the above method, data might still be written to the same Region at the beginning.

TiDB is a database for general usage and does not make assumptions about the data distribution. So it uses only one Region at the beginning to store the data of a table and automatically splits the Region according to the data distribution after real data is inserted.

Given this situation and the need to avoid the hotspot problem, TiDB offers the `Split Region` syntax to optimize performance for the highly-concurrent write-heavy scenario. Based on the above case, now scatter Regions using the `Split Region` syntax and observe the load distribution.

Because the data to be written in the test is entirely discrete within the positive range, you can use the following statement to pre-split the table into 128 Regions within the range of `minInt64` and `maxInt64`:

```
SPLIT TABLE TEST_HOTSPOT BETWEEN (0) AND (9223372036854775807) REGIONS 128;
```

After the pre-split operation, execute the `SHOW TABLE test_hotspot REGIONS;` statement to check the status of Region scattering. If the values of the `SCATTERING` column are all `0`, the scheduling is successful.

You can also check the Region distribution using the table-regions.py script. Currently, the Region distribution is relatively even:

```
[root@172.16.4.4 scripts]# python table-regions.py --host 172.16.4.3 --port
    ↪ 31453 test test_hotspot
[RECORD - test.test_hotspot] - Leaders Distribution:
  total leader count: 127
  store: 1, num_leaders: 21, percentage: 16.54%
  store: 4, num_leaders: 20, percentage: 15.75%
  store: 6, num_leaders: 21, percentage: 16.54%
  store: 46, num_leaders: 21, percentage: 16.54%
  store: 82, num_leaders: 23, percentage: 18.11%
  store: 62, num_leaders: 21, percentage: 16.54%
```

Then operate the write load again:

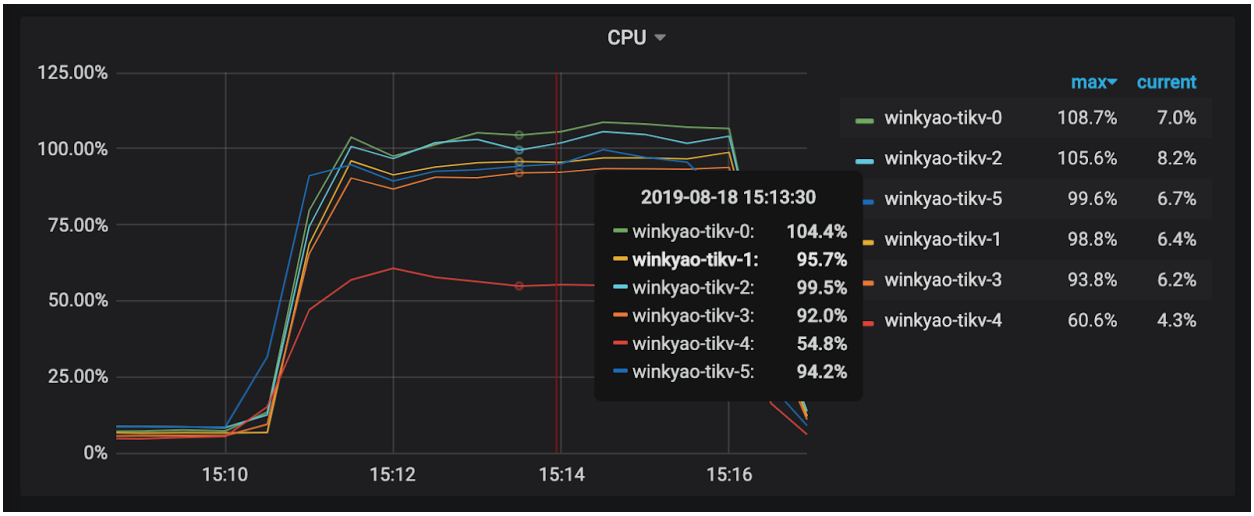

Figure 301: QPS6

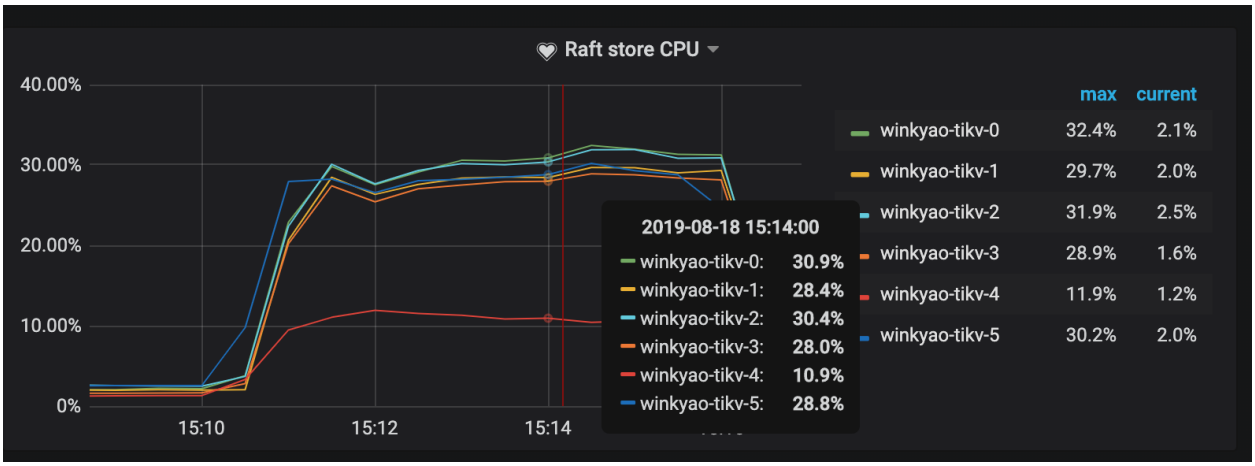

Figure 302: QPS7

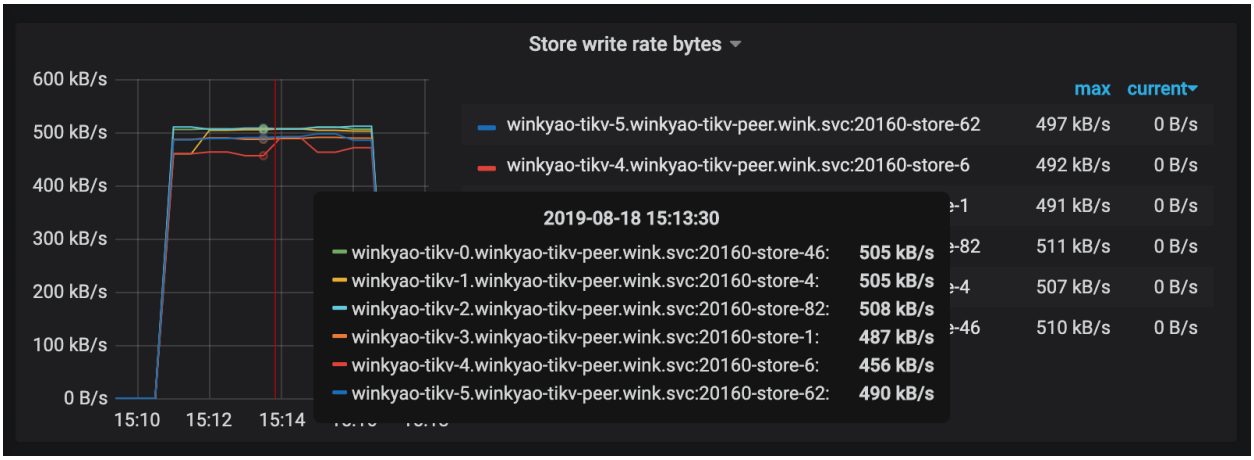

Figure 303: QPS8

You can see that the apparent hotspot problem has been resolved now.

In this case, the table is simple. In other cases, you might also need to consider the hotspot problem of index. For more details on how to pre-split the index Region, refer to Split Region.

4.12.1.7 Complex hotspot problem

If a table does not have a primary key, or the primary key is not the `Int` type and you do not want to generate a randomly distributed primary key ID, TiDB provides an implicit `_tidb_rowid` column as the row ID. Generally, when you do not use the `SHARD_ROW_ID_BITS` parameter, the values of the `_tidb_rowid` column are also monotonically increasing, which might causes hotspots too. Refer to `SHARD_ROW_ID_BITS` description for more details.

To avoid the hotspot problem in this situation, you can use `SHARD_ROW_ID_BITS` and `PRE_SPLIT_REGIONS` when creating a table. For more details about `PRE_SPLIT_REGIONS`, refer to Pre-split Regions.

`SHARD_ROW_ID_BITS` is used to randomly scatter the row ID generated in the `_tidb_rowid` column. `PRE_SPLIT_REGIONS` is used to pre-split the Region after a table is created.

> **Note:**
>
> The value of `PRE_SPLIT_REGIONS` must be smaller than or equal to that of `SHARD_ROW_ID_BITS`.

`tidb_scatter_region` controls whether to wait for Regions to be pre-split and scattered before returning results after the table creation. If there are intensive writes after creating

the table, you need to set the value of this variable to `1`, then TiDB will not return the results to the client until all the Regions are split and scattered. Otherwise, TiDB writes data before the scattering is completed, which will have a significant impact on write performance.

Example:

```
create table t (a int, b int) SHARD_ROW_ID_BITS = 4 PRE_SPLIT_REGIONS=3;
```

- `SHARD_ROW_ID_BITS = 4` means that the values of `tidb_rowid` will be randomly distributed into 16 (16=2^4) ranges.
- `PRE_SPLIT_REGIONS=3` means that the table will be pre-split into 8 (2^3) Regions after it is created.

When data starts to be written into table `t`, the data is written into the pre-split 8 Regions, which avoids the hotspot problem that might be caused if only one Region exists after table creation.

4.12.1.8 Parameter configuration

In v2.1, the latch mechanism is introduced in TiDB to identify transaction conflicts in advance in scenarios where write conflicts frequently appear. The aim is to reduce the retry of transaction commits in TiDB and TiKV caused by write conflicts. Generally, batch tasks use the data already stored in TiDB, so the write conflicts of transaction do not exist. In this situation, you can disable the latch in TiDB to reduce memory allocation for small objects:

```
[txn-local-latches]
enabled = false
```

4.12.2 Best Practices for Using HAProxy in TiDB

This document describes best practices for configuration and usage of HAProxy in TiDB. HAProxy provides load balancing for TCP-based applications. From TiDB clients, you can manipulate data just by connecting to the floating virtual IP address provided by HAProxy, which helps to achieve load balance in the TiDB server layer.

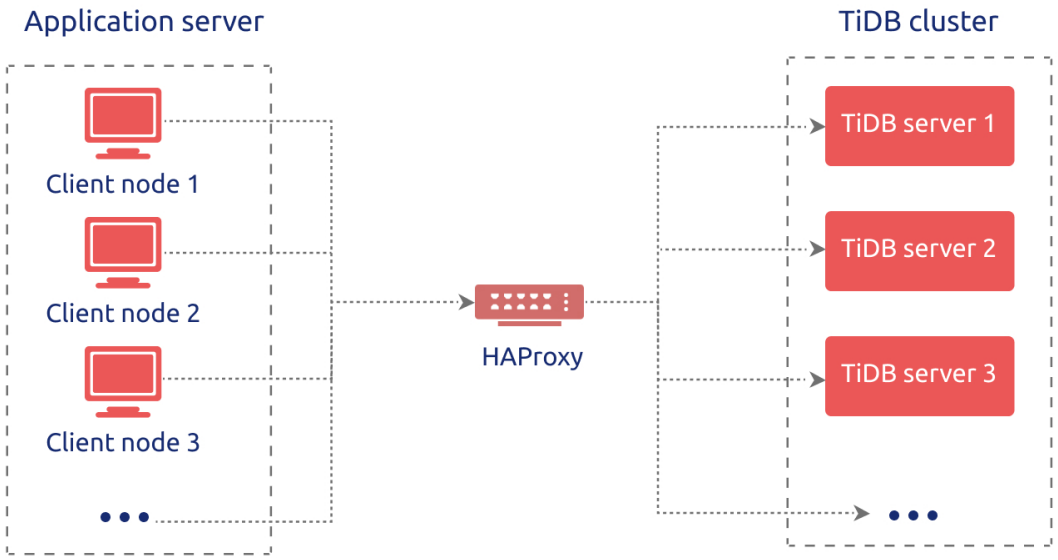

Figure 304: HAProxy Best Practices in TiDB

4.12.2.1 HAProxy overview

HAProxy is free, open-source software written in C language that provides a high availability load balancer and proxy server for TCP and HTTP-based applications. Because of its fast and efficient use of CPU and memory, HAProxy is now widely used by many well-known websites such as GitHub, Bitbucket, Stack Overflow, Reddit, Tumblr, Twitter, Tuenti, and AWS (Amazon Web Services).

HAProxy is written in the year 2000 by Willy Tarreau, the core contributor to the Linux kernel, who is still responsible for the maintenance of the project and provides free software updates in the open-source community. The latest stable version 2.0.0 was released on August 16, 2019, bringing more excellent features.

4.12.2.2 Basic features

- High Availability: HAProxy provides high availability with support for a graceful shutdown and a seamless switchover;
- Load Balancing: Two major proxy modes are supported: TCP, also known as layer 4, and HTTP, also known as layer 7. No less than 9 load balancing algorithms are supported, such as roundrobin, leastconn and random;
- Health Check: HAProxy periodically checks the status of HTTP or TCP mode of the server;
- Sticky Session: HAProxy can stick a client to a specific server for the duration when the application does not support sticky sessions;

- SSL: HTTPS communication and resolution are supported;
- Monitoring and Statistics: Through the web page, you can monitor the service state and traffic flow in real time.

4.12.2.3 Before you begin

Before you deploy HAProxy, make sure that you meet the hardware and software requirements.

4.12.2.3.1 Hardware requirements

For your server, it is recommended to meet the following hardware requirements. You can also improve server specifications according to the load balancing environment.

Hardware resource	Minimum specification
CPU	2 cores, 3.5 GHz
Memory	16 GB
Storage	50 GB (SATA)
Network Interface Card	10G Network Card

4.12.2.3.2 Software requirements

You can use the following operating systems and make sure the required dependencies are installed. If you use yum to install HAProxy, the dependencies are installed along with it and you do not need to separately install them again.

Operating systems

Operating system version	Architecture
Linux 2.4	x86, x86_64, Alpha, SPARC, MIPS, and PA-RISC
Linux 2.6 or 3.x	x86, x86_64, ARM, SPARC, and PPC64
Solaris 8 or 9	UltraSPARC II and III
Solaris 10	Opteron and UltraSPARC
FreeBSD 4.10 ~ 10	x86
OpenBSD 3.1 or later versions	i386, AMD64, macppc, Alpha, and SPARC64
AIX 5.1 ~ 5.3	Power™

Dependencies

- epel-release
- gcc
- systemd-devel

To install the dependencies above, run the following command:

```
yum -y install epel-release gcc systemd-devel
```

4.12.2.4 Deploy HAProxy

You can easily use HAProxy to configure and set up a load-balanced database environment. This section shows general deployment operations. You can customize the configuration file based on your actual scenario.

4.12.2.4.1 Install HAProxy

1. Use yum to install HAProxy:

```
yum -y install haproxy
```

2. Check whether the installation is successful:

```
which haproxy
```

```
/usr/sbin/haproxy
```

HAProxy commands

Execute the following command to print a list of keywords and their basic usage:

```
haproxy --help
```

Option	Description
-v	Reports the version and build date.
-vv	Displays the version, build options, libraries versions and usable pollers.
-d	Enables debug mode.

Option	Description
`-db`	Disables background mode and multi-process mode.
`-dM [<byte>]`	Forces memory poisoning, which means that each and every memory region allocated with malloc() or pool_alloc2() will be filled with `<byte>` before being passed to the caller.
`-V`	Enables verbose mode (disables quiet mode).
`-D`	Starts as a daemon.
`-C <dir>`	Changes to directory `<dir>` before loading configuration files.

Option	Description
`-W`	Master-worker mode.
`-q`	Sets “quiet” mode: This disables some messages during the configuration parsing and during startup.
`-c`	Only performs a check of the configuration files and exits before trying to bind.
`-n <` ↪ `limit` ↪ `>`	Limits the per-process connection limit to `<limit>`.
`-m <` ↪ `limit` ↪ `>`	Limits the total allocatable memory to `<limit>` megabytes across all processes.

Option	Description
`-N <limit>`	Sets the default per-proxy maxconn to `<limit>` instead of the builtin default value (usually 2000).
`-L <name>`	Changes the local peer name to `<name>`, which defaults to the local hostname.
`-p <file>`	Writes all processes' PIDs into `<file>` during startup.
`-de`	Disables the use of epoll(7). epoll(7) is available only on Linux 2.6 and some custom Linux 2.4 systems.
`-dp`	Disables the use of poll(2). select(2) might be used instead.

Option	Description
`-dS`	Disables the use of splice(2), which is broken on older kernels.
`-dR`	Disables SO_REUSEPORT usage.
`-dr`	Ignores server address resolution failures.
`-dV`	Disables SSL verify on the server side.

Option	Description
`-sf <pidlist>`	Sends the "finish" signal to the PIDs in pidlist after startup. The processes which receive this signal wait for all sessions to finish before exiting. This option must be specified last, followed by any number of PIDs. Technically speaking, SIGTTOU and SIGUSR1 are sent.

Option	Description
`-st <` ↪ `pidlist` ↪ `>`	Sends the "terminate" signal to the PIDs in pidlist after startup. The processes which receive this signal terminate immediately, closing all active sessions. This option must be specified last, followed by any number of PIDs. Technically speaking, SIGTTOU and SIGTERM are sent.

Option	Description
`-x <unix_socket>`	Connects to the specified socket and retrieves all the listening sockets from the old process. Then, these sockets are used instead of binding new ones.
`-S <bind>[,<bind_options>...]`	In master-worker mode, creates a master CLI. This CLI enables access to the CLI of every worker. Useful for debugging, it's a convenient way of accessing a leaving process.

For more details on HAProxy command line options, refer to Management Guide of HAProxy and General Commands Manual of HAProxy.

4.12.2.4.2 Configure HAProxy

A configuration template is generated when you use yum to install HAProxy. You can also customize the following configuration items according to your scenario.

```
global                                  # Global configuration.
   log         127.0.0.1 local2         # Global syslog servers (up to two).
   chroot      /var/lib/haproxy         # Changes the current directory and
      ↪ sets superuser privileges for the startup process to improve
      ↪ security.
   pidfile     /var/run/haproxy.pid     # Writes the PIDs of HAProxy processes
      ↪  into this file.
   maxconn     4000                     # The maximum number of concurrent
      ↪ connections for a single HAProxy process.
   user        haproxy                  # Same with the UID parameter.
   group       haproxy                  # Same with the GID parameter. A
      ↪ dedicated user group is recommended.
   nbproc      40                       # The number of processes created when
      ↪  going daemon. When starting multiple processes to forward requests
      ↪ , make sure the value is large enough so that HAProxy does not
      ↪ block processes.
   daemon                               # Makes the process fork into
      ↪ background. It is equivalent to the command line "-D" argument. It
      ↪ can be disabled by the command line "-db" argument.
   stats socket /var/lib/haproxy/stats # The directory where statistics
      ↪ output is saved.

defaults                                # Default configuration.
   log global                           # Inherits the settings of the global
      ↪ configuration.
   retries 2                            # The maximum number of retries to
      ↪ connect to an upstream server. If the number of connection attempts
      ↪  exceeds the value, the backend server is considered unavailable.
   timeout connect 2s                   # The maximum time to wait for a
      ↪ connection attempt to a backend server to succeed. It should be set
      ↪  to a shorter time if the server is located on the same LAN as
      ↪ HAProxy.
   timeout client 30000s                # The maximum inactivity time on the
      ↪ client side.
   timeout server 30000s                # The maximum inactivity time on the
      ↪ server side.

listen admin_stats                      # The name of the Stats page reporting
   ↪  information from frontend and backend. You can customize the name
   ↪ according to your needs.
   bind 0.0.0.0:8080                    # The listening port.
   mode http                            # The monitoring mode.
   option httplog                       # Enables HTTP logging.
   maxconn 10                           # The maximum number of concurrent
```

```
    ↪ connections.
  stats refresh 30s                 # Automatically refreshes the Stats
    ↪ page every 30 seconds.
  stats uri /haproxy                # The URL of the Stats page.
  stats realm HAProxy               # The authentication realm of the
    ↪ Stats page.
  stats auth admin:pingcap123       # User name and password in the Stats
    ↪ page. You can have multiple user names.
  stats hide-version                # Hides the version information of
    ↪ HAProxy on the Stats page.
  stats admin if TRUE               # Manually enables or disables the
    ↪ backend server (supported in HAProxy 1.4.9 or later versions).

listen tidb-cluster                 # Database load balancing.
  bind 0.0.0.0:3390                 # The Floating IP address and
    ↪ listening port.
  mode tcp                          # HAProxy uses layer 4, the transport
    ↪ layer.
  balance leastconn                 # The server with the smallest number
    ↪ of connections receives the connection. `leastconn' is recommended
    ↪ where long sessions are expected, such as LDAP, SQL and TSE, rather
    ↪  than protocols using short sessions, such as HTTP. The algorithm
    ↪ is dynamic, which means that server weights might be adjusted on
    ↪ the fly for slow starts for instance.
  server tidb-1 10.9.18.229:4000 check inter 2000 rise 2 fall 3 # Detects
    ↪ port 4000 at a frequency of once every 2000 milliseconds. If it is
    ↪ detected as successful twice, the server is considered available;
    ↪ if it is detected as failed three times, the server is considered
    ↪ unavailable.
  server tidb-2 10.9.39.208:4000 check inter 2000 rise 2 fall 3
  server tidb-3 10.9.64.166:4000 check inter 2000 rise 2 fall 3
```

4.12.2.4.3 Start HAProxy

There are two methods to start HAProxy.

Method 1: Execute `haproxy`. `/etc/haproxy/haproxy.cfg` is read by default (recommended).

```
haproxy -f /etc/haproxy/haproxy.cfg
```

Method 2: Use `systemd` to start HAProxy.

```
systemctl start haproxy.service
```

4.12.2.4.4 Stop HAProxy

There are two methods to stop HAProxy.

Method 1: Use `kill -9`.

1. Run the following command:

```
ps -ef | grep haproxy
```

2. Terminate the process of HAProxy:

```
kill -9 ${haproxy.pid}
```

Method 2: Use `systemd`.

```
systemctl stop haproxy.service
```

4.12.3 Best Practices for Developing Java Applications with TiDB

This document introduces the best practice for developing Java applications to better use TiDB. Based on some common Java application components that interact with the backend TiDB database, this document also provides the solutions to commonly encountered issues during development.

4.12.3.1 Database-related components in Java applications

Common components that interact with the TiDB database in Java applications include:

- Network protocol: A client interacts with a TiDB server via the standard MySQL protocol.
- JDBC API and JDBC drivers: Java applications usually use the standard JDBC (Java Database Connectivity) API to access a database. To connect to TiDB, you can use a JDBC driver that implements the MySQL protocol via the JDBC API. Such common JDBC drivers for MySQL include MySQL Connector/J and MariaDB Connector/J.
- Database connection pool: To reduce the overhead of creating a connection each time it is requested, applications usually use a connection pool to cache and reuse connections. JDBC DataSource defines a connection pool API. You can choose from different open-source connection pool implementations as needed.
- Data access framework: Applications usually use a data access framework such as MyBatis and Hibernate to further simplify and manage the database access operations.
- Application implementation: The application logic controls when to send what commands to the database. Some applications use Spring Transaction aspects to manage transactions' start and commit logics.

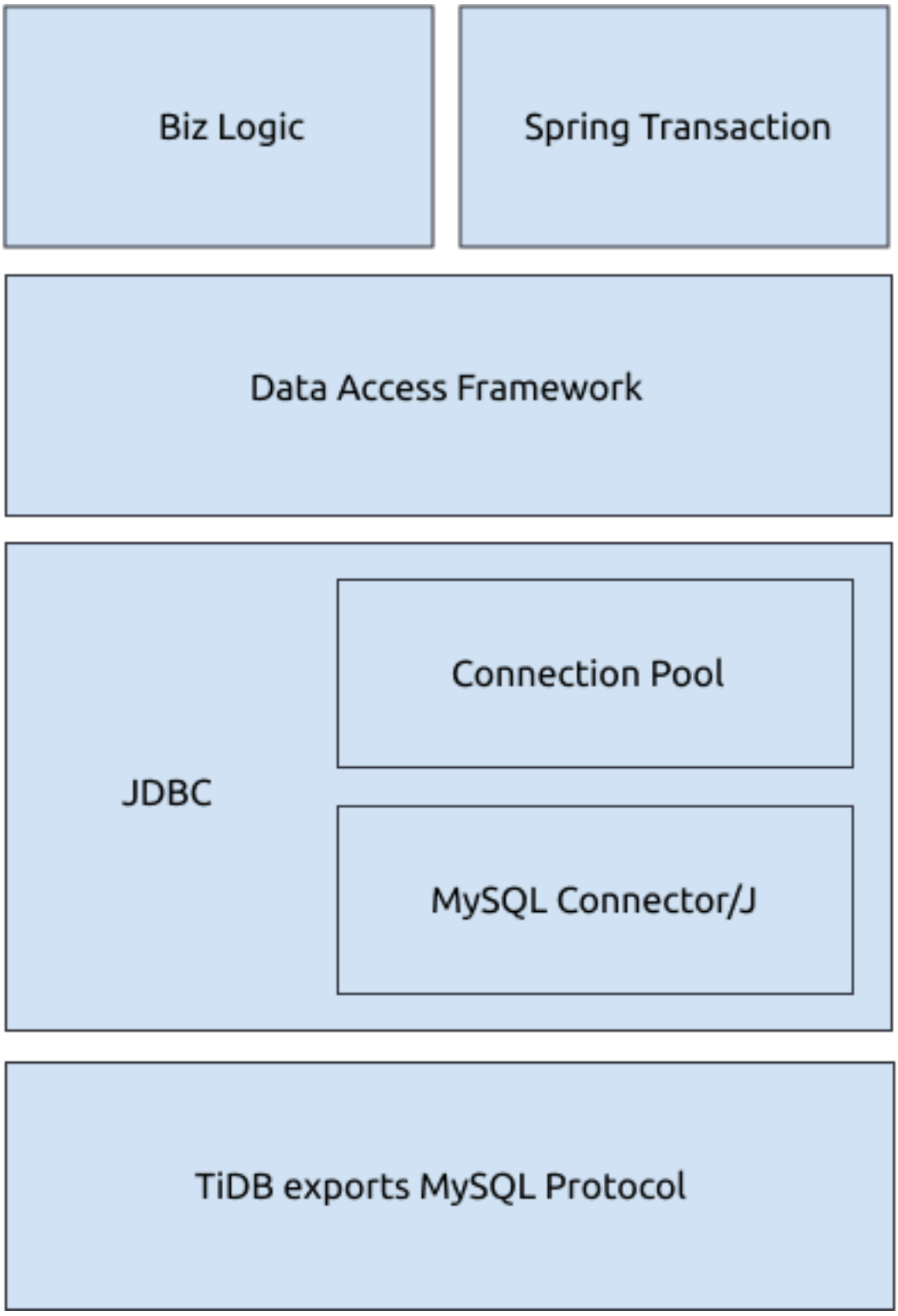

Figure 305: Java application components

From the above diagram, you can see that a Java application might do the following things:

- Implement the MySQL protocol via the JDBC API to interact with TiDB.
- Get a persistent connection from the connection pool.
- Use a data access framework such as MyBatis to generate and execute SQL statements.
- Use Spring Transaction to automatically start or stop a transaction.

The rest of this document describes the issues and their solutions when you develop a Java application using the above components.

4.12.3.2 JDBC

Java applications can be encapsulated with various frameworks. In most of the frameworks, JDBC API is called on the bottommost level to interact with the database server. For JDBC, it is recommended that you focus on the following things:

- JDBC API usage choice
- API Implementer's parameter configuration

4.12.3.2.1 JDBC API

For JDBC API usage, see JDBC official tutorial. This section covers the usage of several important APIs.

Use Prepare API

For OLTP (Online Transactional Processing) scenarios, the SQL statements sent by the program to the database are several types that can be exhausted after removing parameter changes. Therefore, it is recommended to use Prepared Statements instead of regular execution from a text file and reuse Prepared Statements to execute directly. This avoids the overhead of repeatedly parsing and generating SQL execution plans in TiDB.

At present, most upper-level frameworks call the Prepare API for SQL execution. If you use the JDBC API directly for development, pay attention to choosing the Prepare API.

In addition, with the default implementation of MySQL Connector/J, only client-side statements are preprocessed, and the statements are sent to the server in a text file after `?` is replaced on the client. Therefore, in addition to using the Prepare API, you also need to configure `useServerPrepStmts = true` in JDBC connection parameters before you perform statement preprocessing on the TiDB server. For detailed parameter configuration, see MySQL JDBC parameters.

Use Batch API

For batch inserts, you can use the `addBatch/executeBatch` API. The `addBatch()` method is used to cache multiple SQL statements first on the client, and then send them to the database server together when calling the `executeBatch` method.

Note:

In the default MySQL Connector/J implementation, the sending time of the SQL statements that are added to batch with `addBatch()` is delayed to the time when `executeBatch()` is called, but the statements will still be sent one by one during the actual network transfer. Therefore, this method usually does not reduce the amount of communication overhead.

If you want to batch network transfer, you need to configure `rewriteBatchedStatements = true` in the JDBC connection parameters. For the detailed parameter configuration, see Batch-related parameters.

Use `StreamingResult` to get the execution result

In most scenarios, to improve execution efficiency, JDBC obtains query results in advance and save them in client memory by default. But when the query returns a super large result set, the client often wants the database server to reduce the number of records returned at a time, and waits until the client's memory is ready and it requests for the next batch.

Usually, there are two kinds of processing methods in JDBC:

- Set `FetchSize` to `Integer.MIN_VALUE` to ensure that the client does not cache. The client will read the execution result from the network connection through `StreamingResult`.
- To use Cursor Fetch, first set `FetchSize` as a positive integer and configure `useCursorFetch=true` in the JDBC URL.

TiDB supports both methods, but it is preferred that you use the first method, because it is a simpler implementation and has a better execution efficiency.

4.12.3.2.2 MySQL JDBC parameters

JDBC usually provides implementation-related configurations in the form of JDBC URL parameters. This section introduces MySQL Connector/J's parameter configurations (If you use MariaDB, see MariaDB's parameter configurations). Because this document cannot cover all configuration items, it mainly focuses on several parameters that might affect performance.

Prepare-related parameters

This section introduces parameters related to `Prepare`.

`useServerPrepStmts`

`useServerPrepStmts` is set to `false` by default, that is, even if you use the Prepare API, the "prepare" operation will be done only on the client. To avoid the parsing overhead of the

server, if the same SQL statement uses the Prepare API multiple times, it is recommended to set this configuration to `true`.

To verify that this setting already takes effect, you can do:

- Go to TiDB monitoring dashboard and view the request command type through **Query Summary > QPS By Instance**.
- If `COM_QUERY` is replaced by `COM_STMT_EXECUTE` or `COM_STMT_PREPARE` in the request, it means this setting already takes effect.

`cachePrepStmts`

Although `useServerPrepStmts=true` allows the server to execute Prepared Statements, by default, the client closes the Prepared Statements after each execution and does not reuse them. This means that the "prepare" operation is not even as efficient as text file execution. To solve this, it is recommended that after setting `useServerPrepStmts=true`, you should also configure `cachePrepStmts=true`. This allows the client to cache Prepared Statements.

To verify that this setting already takes effect, you can do:

- Go to TiDB monitoring dashboard and view the request command type through **Query Summary > QPS By Instance**.
- If the number of `COM_STMT_EXECUTE` in the request is far more than the number of `COM_STMT_PREPARE`, it means this setting already takes effect.

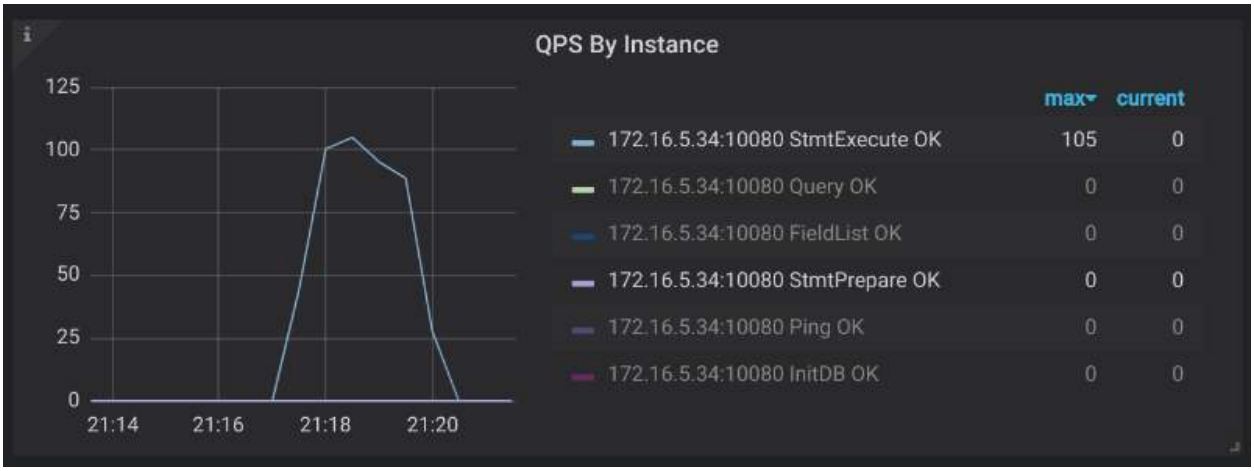

Figure 306: QPS By Instance

In addition, configuring `useConfigs=maxPerformance` will configure multiple parameters at the same time, including `cachePrepStmts=true`.

`prepStmtCacheSqlLimit`

After configuring `cachePrepStmts`, also pay attention to the `prepStmtCacheSqlLimit` configuration (the default value is `256`). This configuration controls the maximum length of the Prepared Statements cached on the client.

The Prepared Statements that exceed this maximum length will not be cached, so they cannot be reused. In this case, you may consider increasing the value of this configuration depending on the actual SQL length of the application.

You need to check whether this setting is too small if you:

- Go to TiDB monitoring dashboard and view the request command type through **Query Summary > QPS By Instance**.
- And find that `cachePrepStmts=true` has been configured, but `COM_STMT_PREPARE` is still mostly equal to `COM_STMT_EXECUTE` and `COM_STMT_CLOSE` exists.

`prepStmtCacheSize`

`prepStmtCacheSize` controls the number of cached Prepared Statements (the default value is `25`). If your application requires "preparing" many types of SQL statements and wants to reuse Prepared Statements, you can increase this value.

To verify that this setting already takes effect, you can do:

- Go to TiDB monitoring dashboard and view the request command type through **Query Summary > QPS By Instance**.
- If the number of `COM_STMT_EXECUTE` in the request is far more than the number of `COM_STMT_PREPARE`, it means this setting already takes effect.

Batch-related parameters

While processing batch writes, it is recommended to configure `rewriteBatchedStatements=true`. After using `addBatch()` or `executeBatch()`, JDBC still sends SQL one by one by default, for example:

```
pstmt = prepare( “insert into t (a) values(?)” );
pstmt.setInt(1, 10);
pstmt.addBatch();
pstmt.setInt(1, 11);
pstmt.addBatch();
pstmt.setInt(1, 12);
pstmt.executeBatch();
```

Although `Batch` methods are used, the SQL statements sent to TiDB are still individual `INSERT` statements:

```
insert into t(a) values(10);
insert into t(a) values(11);
insert into t(a) values(12);
```

But if you set `rewriteBatchedStatements=true`, the SQL statements sent to TiDB will be a single `INSERT` statement:

```
insert into t(a) values(10),(11),(12);
```

Note that the rewrite of the `INSERT` statements is to concatenate the values after multiple "values" keywords into a whole SQL statement. If the `INSERT` statements have other differences, they cannot be rewritten, for example:

```
insert into t (a) values (10) on duplicate key update a = 10;
insert into t (a) values (11) on duplicate key update a = 11;
insert into t (a) values (12) on duplicate key update a = 12;
```

The above `INSERT` statements cannot be rewritten into one statement. But if you change the three statements into the following ones:

```
insert into t (a) values (10) on duplicate key update a = values(a);
insert into t (a) values (11) on duplicate key update a = values(a);
insert into t (a) values (12) on duplicate key update a = values(a);
```

Then they meet the rewrite requirement. The above `INSERT` statements will be rewritten into the following one statement:

```
insert into t (a) values (10), (11), (12) on duplicate key update a =
    ↪ values(a);
```

If there are three or more updates during the batch update, the SQL statements will be rewritten and sent as multiple queries. This effectively reduces the client-to-server request overhead, but the side effect is that a larger SQL statement is generated. For example:

```
update t set a = 10 where id = 1; update t set a = 11 where id = 2; update
    ↪ t set a = 12 where id = 3;
```

In addition, because of a client bug, if you want to configure `rewriteBatchedStatements` ↪ `=true` and `useServerPrepStmts=true` during batch update, it is recommended that you also configure the `allowMultiQueries=true` parameter to avoid this bug.

Check parameters before execution

Through monitoring, you might notice that although the application only performs `INSERT` operations to the TiDB cluster, there are a lot of redundant `SELECT` statements. Usually this happens because JDBC sends some SQL statements to query the settings, for example, `select @@session.transaction_read_only`. These SQL statements are useless for TiDB, so it is recommended that you configure `useConfigs=maxPerformance` to avoid extra overhead.

`useConfigs=maxPerformance` configuration includes a group of configurations:

```
cacheServerConfiguration=true
useLocalSessionState=true
```

```
elideSetAutoCommits=true
alwaysSendSetIsolation=false
enableQueryTimeouts=false
```

After it is configured, you can check the monitoring to see a decreased number of `SELECT` statements.

4.12.3.3 Connection pool

Building TiDB (MySQL) connections is relatively expensive (for OLTP scenarios at least), because in addition to building a TCP connection, connection authentication is also required. Therefore, the client usually saves the TiDB (MySQL) connections to the connection pool for reuse.

Java has many connection pool implementations such as HikariCP, tomcat-jdbc, durid, c3p0, and dbcp. TiDB does not limit which connection pool you use, so you can choose whichever you like for your application.

4.12.3.3.1 Configure the number of connections

It is a common practice that the connection pool size is well adjusted according to the application's own needs. Take HikariCP as an example:

- `maximumPoolSize`: The maximum number of connections in the connection pool. If this value is too large, TiDB consumes resources to maintain useless connections. If this value is too small, the application gets slow connections. So configure this value for your own good. For details, see About Pool Sizing.
- `minimumIdle`: The minimum number of idle connections in the connection pool. It is mainly used to reserve some connections to respond to sudden requests when the application is idle. You can also configure it according to your application needs.

The application needs to return the connection after finishing using it. It is also recommended that the application use the corresponding connection pool monitoring (such as `metricRegistry`) to locate the connection pool issue in time.

4.12.3.3.2 Probe configuration

The connection pool maintains persistent connections to TiDB. TiDB does not proactively close client connections by default (unless an error is reported), but generally there will be network proxies such as LVS or HAProxy between the client and TiDB. Usually, these proxies will proactively clean up connections that are idle for a certain period of time. In addition to paying attention to the idle configuration of the proxies, the connection pool also needs to keep alive or probe connections.

If you often see the following error in your Java application:

```
The last packet sent successfully to the server was 3600000 milliseconds ago
    ↪ . The driver has not received any packets from the server. com.mysql.
    ↪ jdbc.exceptions.jdbc4.CommunicationsException: Communications link
    ↪ failure
```

If `n` in `n milliseconds ago` is `0` or a very small value, it is usually because the executed SQL operation causes TiDB to exit abnormally. To find the cause, it is recommended to check the TiDB stderr log.

If `n` is a very large value (such as `3600000` in the above example), it is likely that this connection was idle for a long time and then closed by the intermediate proxy. The usual solution is to increase the value of the proxy's idle configuration and allow the connection pool to:

- Check whether the connection is available before using the connection every time
- Regularly check whether the connection is available using a separate thread.
- Send a test query regularly to keep alive connections

Different connection pool implementations might support one or more of the above methods. You can check your connection pool documentation to find the corresponding configuration.

4.12.3.4 Data access framework

Applications often use some kind of data access framework to simplify database access.

4.12.3.4.1 MyBatis

MyBatis is a popular Java data access framework. It is mainly used to manage SQL queries and complete the mapping between result sets and Java objects. MyBatis is highly compatible with TiDB. MyBatis rarely has problems based on its historical issues.

Here this document mainly focuses on the following configurations.

Mapper parameters

MyBatis Mapper supports two parameters:

- `select 1 from t where id = #{param1}` will be converted to `select 1 from t where id =?` as a Prepared Statement and be "prepared", and the actual parameter will be used for reuse. You can get the best performance when using this parameter with the previously mentioned Prepare connection parameters.
- `select 1 from t where id = ${param2}` will be replaced with `select 1 from t where id = 1` as a text file and be executed. If this statement is replaced with different parameters and is executed, MyBatis will send different requests for "preparing" the statements to TiDB. This might cause TiDB to cache a large number of Prepared Statements, and executing SQL operations this way has injection security risks.

Dynamic SQL Batch

Dynamic SQL - foreach

To support the automatic rewriting of multiple `INSERT` statements into the form of `insert ... values(...), (...), ...`, in addition to configuring `rewriteBatchedStatements =true` in JDBC as mentioned before, MyBatis can also use dynamic SQL to semi-automatically generate batch inserts. Take the following mapper as an example:

```
<insert id="insertTestBatch" parameterType="java.util.List" fetchSize="1">
  insert into test
   (id, v1, v2)
  values
  <foreach item="item" index="index" collection="list" separator=",">
  (
   #{item.id}, #{item.v1}, #{item.v2}
  )
  </foreach>
  on duplicate key update v2 = v1 + values(v1)
</insert>
```

This mapper generates an `insert on duplicate key update` statement. The number of `(?,?,?)` following "values" is determined by the number of passed lists. Its final effect is similar to using `rewriteBatchStatements=true`, which also effectively reduces communication overhead between the client and TiDB.

As mentioned before, you also need to note that the Prepared Statements will not be cached after their maximum length exceeds the value of `prepStmtCacheSqlLimit`.

Streaming result

A previous section introduces how to stream read execution results in JDBC. In addition to the corresponding configurations of JDBC, if you want to read a super large result set in MyBatis, you also need to note that:

- You can set `fetchSize` for a single SQL statement in the mapper configuration (see the previous code block). Its effect is equivalent to calling `setFetchSize` in JDBC.
- You can use the query interface with `ResultHandler` to avoid getting the entire result set at once.
- You can use the `Cursor` class for stream reading.

If you configure mappings using XML, you can stream read results by configuring `fetchSize="-2147483648"(Integer.MIN_VALUE)` in the mapping's `<select>` section.

```
<select id="getAll" resultMap="postResultMap" fetchSize="-2147483648">
  select * from post;
</select>
```

If you configure mappings using code, you can add the `@Options(fetchSize = Integer.MIN_VALUE)` annotation and keep the type of results as `Cursor` so that the SQL results can be read in streaming.

```
@Select("select * from post")
@Options(fetchSize = Integer.MIN_VALUE)
Cursor<Post> queryAllPost();
```

4.12.3.4.2 ExecutorType

You can choose `ExecutorType` during `openSession`. MyBatis supports three types of executors:

- Simple: The Prepared Statements are called to JDBC for each execution (if the JDBC configuration item `cachePrepStmts` is enabled, repeated Prepared Statements will be reused)
- Reuse: The Prepared Statements are cached in `executor`, so that you can reduce duplicate calls for Prepared Statements without using the JDBC `cachePrepStmts`
- Batch: Each update operation (`INSERT`/`DELETE`/`UPDATE`) will first be added to the batch, and will be executed until the transaction commits or a `SELECT` query is performed. If `rewriteBatchStatements` is enabled in the JDBC layer, it will try to rewrite the statements. If not, the statements will be sent one by one.

Usually, the default value of `ExecutorType` is `Simple`. You need to change `ExecutorType` when calling `openSession`. If it is the batch execution, you might find that in a transaction the `UPDATE` or `INSERT` statements are executed pretty fast, but it is slower when reading data or committing the transaction. This is actually normal, so you need to note this when troubleshooting slow SQL queries.

4.12.3.5 Spring Transaction

In the real world, applications might use Spring Transaction and AOP aspects to start and stop transactions.

By adding the `@Transactional` annotation to the method definition, AOP starts the transaction before the method is called, and commits the transaction before the method returns the result. If your application has a similar need, you can find `@Transactional` in code to determine when the transaction is started and closed.

Pay attention to a special case of embedding. If it occurs, Spring will behave differently based on the Propagation configuration. Because TiDB does not support savepoint, nested transactions are not supported yet.

4.12.3.6 Misc

This section introduces some useful tools for Java to help you troubleshoot issues.

4.12.3.6.1 Troubleshooting tools

Using the powerful troubleshooting tools of JVM is recommended when an issue occurs in your Java application and you do not know the application logic. Here are a few common tools:

jstack

jstack is similar to pprof/goroutine in Go, which can easily troubleshoot the process stuck issue.

By executing `jstack pid`, you can output the IDs and stack information of all threads in the target process. By default, only the Java stack is output. If you want to output the C++ stack in the JVM at the same time, add the `-m` option.

By using jstack multiple times, you can easily locate the stuck issue (for example, a slow query from application's view due to using Batch ExecutorType in Mybatis) or the application deadlock issue (for example, the application does not send any SQL statement because it is preempting a lock before sending it).

In addition, `top -p $ PID -H` or Java swiss knife are common methods to view the thread ID. Also, to locate the issue of "a thread occupies a lot of CPU resources and I don't know what it is executing", do the following steps:

- Use `printf "%x\n" pid` to convert the thread ID to hexadecimal.
- Go to the jstack output to find the stack information of the corresponding thread.

jmap & mat

Unlike pprof/heap in Go, jmap dumps the memory snapshot of the entire process (in Go, it is the sampling of the distributor), and then the snapshot can be analyzed by another tool mat.

Through mat, you can see the associated information and attributes of all objects in the process, and you can also observe the running status of the thread. For example, you can use mat to find out how many MySQL connection objects exist in the current application, and what is the address and status information of each connection object.

Note that mat only handles reachable objects by default. If you want to troubleshoot young GC issues, you can adjust mat configuration to view unreachable objects. In addition, for investigating the memory allocation of young GC issues (or a large number of short-lived objects), using Java Flight Recorder is more convenient.

trace

Online applications usually do not support modifying the code, but it is often desired that dynamic instrumentation is performed in Java to locate issues. Therefore, using btrace or arthas trace is a good option. They can dynamically insert trace code without restarting the application process.

Flame graph

Obtaining flame graphs in Java applications is tedious. For details, see Java Flame Graphs Introduction: Fire For Everyone!.

4.12.3.7 Conclusion

Based on commonly used Java components that interact with databases, this document describes the common problems and solutions for developing Java applications with TiDB. TiDB is highly compatible with the MySQL protocol, so most of the best practices for MySQL-based Java applications also apply to TiDB.

Join us at TiDB Community slack channel, and share with broad TiDB user group about your experience or problems when you develop Java applications with TiDB.

4.12.4 PD Scheduling Best Practices

This document details the principles and strategies of PD scheduling through common scenarios to facilitate your application. This document assumes that you have a basic understanding of TiDB, TiKV and PD with the following core concepts:

- leader/follower/learner
- operator
- operator step
- pending/down
- region/peer/Raft group
- region split
- scheduler
- store

> **Note:**
>
> This document initially targets TiDB 3.0. Although some features are not supported in earlier versions (2.x), the underlying mechanisms are similar and this document can still be used as a reference.

4.12.4.1 PD scheduling policies

This section introduces the principles and processes involved in the scheduling system.

4.12.4.1.1 Scheduling process

The scheduling process generally has three steps:

1. Collect information

   Each TiKV node periodically reports two types of heartbeats to PD:

- `StoreHeartbeat`: Contains the overall information of stores, including disk capacity, available storage, and read/write traffic
- `RegionHeartbeat`: Contains the overall information of regions, including the range of each region, peer distribution, peer status, data volume, and read/write traffic

PD collects and restores this information for scheduling decisions.

2. Generate operators

   Different schedulers generate the operators based on their own logic and requirements, with the following considerations:

   - Do not add peers to a store in abnormal states (disconnected, down, busy, low space)
   - Do not balance regions in abnormal states
   - Do not transfer a leader to a pending peer
   - Do not remove a leader directly
   - Do not break the physical isolation of various region peers
   - Do not violate constraints such as label property

3. Execute operators

   To execute the operators, the general procedure is:

   1. The generated operator first joins a queue managed by `OperatorController`.
   2. `OperatorController` takes the operator out of the queue and executes it with a certain amount of concurrency based on the configuration. This step is to assign each operator step to the corresponding region leader.
   3. The operator is marked as "finish" or "timeout" and removed from the queue.

4.12.4.1.2 Load balancing

Region primarily relies on `balance-leader` and `balance-region` schedulers to achieve load balance. Both schedulers target distributing regions evenly across all stores in the cluster but with separate focuses: `balance-leader` deals with region leader to balance incoming client requests, whereas `balance-region` concerns itself with each region peer to redistribute the pressure of storage and avoid exceptions like out of storage space.

`balance-leader` and `balance-region` share a similar scheduling process:

1. Rate stores according to their resource availability.
2. `balance-leader` or `balance-region` constantly transfer leaders or peers from stores with high scores to those with low scores.

However, their rating methods are different. `balance-leader` uses the sum of all region sizes corresponding to leaders in a store, whereas the way of `balance-region` is relatively complicated. Depending on the specific storage capacity of each node, the rating method of `balance-region` might:

- based on the amount of data when there is sufficient storage (to balance data distribution among nodes).
- based on the available storage when there is insufficient storage (to balance the storage availability on different nodes).
- based on the weighted sum of the two factors above when neither of the situations applies.

Because different nodes might differ in performance, you can also set the weight of load balancing for different stores. `leader-weight` and `region-weight` are used to control the leader weight and region weight respectively ("1" by default for both). For example, when the `leader-weight` of a store is set to "2", the number of leaders on the node is about twice as many as that of other nodes after the scheduling stabilizes. Similarly, when the `leader-weight` of a store is set to "0.5", the number of leaders on the node is about half as many as that of other nodes.

4.12.4.1.3 Hot regions scheduling

For hot regions scheduling, use `hot-region-scheduler`. Currently in TiDB 3.0, the process is performed as follows:

1. Count hot regions by determining read/write traffic that exceeds a certain threshold for a certain period based on the information reported by stores.
2. Redistribute these regions in a similar way to load balancing.

For hot write regions, `hot-region-scheduler` attempts to redistribute both region peers and leaders; for hot read regions, `hot-region-scheduler` only redistributes region leaders.

4.12.4.1.4 Cluster topology awareness

Cluster topology awareness enables PD to distribute replicas of a region as much as possible. This is how TiKV ensures high availability and disaster recovery capability. PD continuously scans all regions in the background. When PD finds that the distribution of regions is not optimal, it generates an operator to replace peers and redistribute regions.

The component to check region distribution is `replicaChecker`, which is similar to a scheduler except that it cannot be disabled. `replicaChecker` schedules based on the the configuration of `location-labels`. For example, `[zone,rack,host]` defines a three-tier topology for a cluster. PD attempts to schedule region peers to different zones first, or to different racks when zones are insufficient (for example, 2 zones for 3 replicas), or to different hosts when racks are insufficient, and so on.

4.12.4.1.5 Scale-down and failure recovery

Scale-down refers to the process when you take a store offline and mark it as "offline" using a command. PD replicates the regions on the offline node to other nodes by scheduling. Failure recovery applies when stores failed and cannot be recovered. In this case, regions with peers distributed on the corresponding store might lose replicas, which requires PD to replenish on other nodes.

The processes of scale-down and failure recovery are basically the same. `replicaChecker` ↪ finds a region peer in abnormal states, and then generates an operator to replace the abnormal peer with a new one on a healthy store.

4.12.4.1.6 Region merge

Region merge refers to the process of merging adjacent small regions. It serves to avoid unnecessary resource consumption by a large number of small or even empty regions after data deletion. Region merge is performed by `mergeChecker`, which processes in a similar way to `replicaChecker`: PD continuously scans all regions in the background, and generates an operator when contiguous small regions are found.

4.12.4.2 Query scheduling status

You can check the status of scheduling system through metrics, pd-ctl and logs. This section briefly introduces the methods of metrics and pd-ctl. Refer to PD Monitoring Metrics and PD Control for details.

4.12.4.2.1 Operator status

The **Grafana PD/Operator** page shows the metrics about operators, among which:

- Schedule operator create: Operator creating information
- Operator finish duration: Execution time consumed by each operator
- Operator step duration: Execution time consumed by the operator step

You can query operators using pd-ctl with the following commands:

- `operator show`: Queries all operators generated in the current scheduling task
- `operator show [admin | leader | region]`: Queries operators by type

4.12.4.2.2 Balance status

The **Grafana PD/Statistics - Balance** page shows the metrics about load balancing, among which:

- Store leader/region score: Score of each store
- Store leader/region count: The number of leaders/regions in each store
- Store available: Available storage on each store

You can use store commands of pd-ctl to query balance status of each store.

4.12.4.2.3 Hot Region status

The **Grafana PD/Statistics - hotspot** page shows the metrics about hot regions, among which:

- Hot write region's leader/peer distribution: the leader/peer distribution in hot write regions
- Hot read region's leader distribution: the leader distribution in hot read regions

You can also query the status of hot regions using pd-ctl with the following commands:

- `hot read`: Queries hot read regions
- `hot write`: Queries hot write regions
- `hot store`: Queries the distribution of hot regions by store
- `region topread [limit]`: Queries the region with top read traffic
- `region topwrite [limit]`: Queries the region with top write traffic

4.12.4.2.4 Region health

The **Grafana PD/Cluster/Region health** panel shows the metrics about regions in abnormal states.

You can query the list of regions in abnormal states using pd-ctl with region check commands:

- `region check miss-peer`: Queries regions without enough peers
- `region check extra-peer`: Queries regions with extra peers
- `region check down-peer`: Queries regions with down peers
- `region check pending-peer`: Queries regions with pending peers

4.12.4.3 Control scheduling strategy

You can use pd-ctl to adjust the scheduling strategy from the following three aspects. Refer to PD Control for more details.

4.12.4.3.1 Add/delete scheduler manually

PD supports dynamically adding and removing schedulers directly through pd-ctl. For example:

- `scheduler show`: Shows currently running schedulers in the system
- `scheduler remove balance-leader-scheduler`: Removes (disable) balance-leader-scheduler
- `scheduler add evict-leader-scheduler 1`: Adds a scheduler to remove all leaders in Store 1

4.12.4.3.2 Add/delete Operators manually

PD also supports adding or removing operators directly through pd-ctl. For example:

- `operator add add-peer 2 5`: Adds peers to Region 2 in Store 5
- `operator add transfer-leader 2 5`: Migrates the leader of Region 2 to Store 5
- `operator add split-region 2`: Splits Region 2 into two regions evenly in size
- `operator remove 2`: Removes currently pending operator in Region 2

4.12.4.3.3 Adjust scheduling parameter

You can check the scheduling configuration using the `config show` command in pd-ctl, and adjust the values using `config set {key} {value}`. Common adjustments include:

- `leader-schedule-limit`: Controls the concurrency of transferring leader scheduling
- `region-schedule-limit`: Controls the concurrency of adding/deleting peer scheduling
- `disable-replace-offline-replica`: Determines whether to disable the scheduling to take nodes offline
- `disable-location-replacement`: Determines whether to disable the scheduling that handles the isolation level of regions
- `max-snapshot-count`: Controls the maximum concurrency of sending/receiving snapshots for each store

4.12.4.4 PD scheduling in common scenarios

This section illustrates the best practices of PD scheduling strategies through several typical scenarios.

4.12.4.4.1 Leaders/regions are not evenly distributed

The rating mechanism of PD determines that leader count and region count of different stores cannot fully reflect the load balancing status. Therefore, it is necessary to confirm whether there is load imbalancing from the actual load of TiKV or storage usage.

Once you have confirmed that leaders/region are not evenly distributed, you need to check the rating of different stores.

If the scores of different stores are close, it means PD mistakenly believes that leaders/regions are evenly distributed. Possible reasons are:

- There are hot regions that cause load imbalancing. In this case, you need to analyze further based on hot regions scheduling.
- There are a large number of empty regions or small regions, which leads to a great difference in the number of leaders in different stores and high pressure on Raft store. This is the time for a region merge scheduling.

- Hardware and software environment varies among stores. You can adjust the values of `leader-weight` and `region-weight` accordingly to control the distribution of leader/region.
- Other unknown reasons. Still you can adjust the values of `leader-weight` and `region-weight` to control the distribution of leader/region.

If there is a big difference in the rating of different stores, you need to examine the operator-related metrics, with special focus on the generation and execution of operators. There are two main situations:

- When operators are generated normally but the scheduling process is slow, it is possible that:
  - The scheduling speed is limited by default for load balancing purpose. You can adjust `leader-schedule-limit` or `region-schedule-limit` to larger values without significantly impacting regular services. In addition, you can also properly ease the restrictions specified by `max-pending-peer-count` and `max-snapshot-count`.
  - Other scheduling tasks are running concurrently, which slows down the balancing. In this case, if the balancing takes precedence over other scheduling tasks, you can stop other tasks or limit their speeds. For example, if you take some nodes offline when balancing is in progress, both operations consume the quota of `region-schedule-limit`. In this case, you can limit the speed of scheduler to remove nodes, or simply set `disable-replace-offline-replica = true` to temporarily disable it.
  - The scheduling process is too slow. You can check the **Operator step duration** metric to confirm the cause. Generally, steps that do not involve sending and receiving snapshots (such as `TransferLeader`, `RemovePeer`, `PromoteLearner`) should be completed in milliseconds, while steps that involve snapshots (such as `AddLearner` and `AddPeer`) are expected to be completed in tens of seconds. If the duration is obviously too long, it could be caused by high pressure on TiKV or bottleneck in network, etc., which needs specific analysis.
- PD fails to generate the corresponding balancing scheduler. Possible reasons include:
  - The scheduler is not activated. For example, the corresponding scheduler is deleted, or its limit it set to "0".
  - Other constraints. For example, `evict-leader-scheduler` in the system prevents leaders from being migrating to the corresponding store. Or label property is set, which makes some stores reject leaders.
  - Restrictions from the cluster topology. For example, in a cluster of 3 replicas across 3 data centers, 3 replicas of each region are distributed in different data centers due to replica isolation. If the number of stores is different among these data centers, the scheduling can only reach a balanced state within each data center, but not balanced globally.

4.12.4.4.2 Taking nodes offline is slow

This scenario requires examining the generation and execution of operators through related metrics.

If operators are successfully generated but the scheduling process is slow, possible reasons are:

- The scheduling speed is limited by default. You can adjust `leader-schedule-limit` or `replica-schedule-limit` to larger value.s Similarly, you can consider loosening the limits on `max-pending-peer-count` and `max-snapshot-count`.
- Other scheduling tasks are running concurrently and racing for resources in the system. You can refer to the solution in Leaders/regions are not evenly distributed.
- When you take a single node offline, a number of region leaders to be processed (around 1/3 under the configuration of 3 replicas) are distributed on the node to remove. Therefore, the speed is limited by the speed at which snapshots are generated by this single node. You can speed it up by manually adding an `evict-leader-scheduler` to migrate leaders.

If the corresponding operator fails to generate, possible reasons are:

- The operator is stopped, or `replica-schedule-limit` is set to "0".
- There is no proper node for region migration. For example, if the available capacity size of the replacing node (of the same label) is less than 20%, PD will stop scheduling to avoid running out of storage on that node. In such case, you need to add more nodes or delete some data to free the space.

4.12.4.4.3 Bringing nodes online is slow

Currently, bringing nodes online is scheduled through the balance region mechanism. You can refer to Leaders/regions are not evenly distributed for troubleshooting.

4.12.4.4.4 Hot regions are not evenly distributed

Hot regions scheduling issues generally fall into the following categories:

- Hot regions can be observed via PD metrics, but the scheduling speed cannot keep up to redistribute hot regions in time.

  Solution: adjust `hot-region-schedule-limit` to a larger value, and reduce the limit quota of other schedulers to speed up hot regions scheduling. Or you can adjust `hot-region-cache-hits-threshold` to a smaller value to make PD more sensitive to traffic changes.

- Hotspot formed on a single region. For example, a small table is intensively scanned by a massive amount of requests. This can also be detected from PD metrics. Because you cannot actually distribute a single hotspot, you need to manually add a `split-region` operator to split such a region.

- The load of some nodes is significantly higher than that of other nodes from TiKV-related metrics, which becomes the bottleneck of the whole system. Currently, PD counts hotspots through traffic analysis only, so it is possible that PD fails to identify hotspots in certain scenarios. For example, when there are intensive point lookup requests for some regions, it might not be obvious to detect in traffic, but still the high QPS might lead to bottlenecks in key modules.

  Solutions: Firstly, locate the table where hot regions are formed based on the specific business. Then add a `scatter-range-scheduler` scheduler to make all regions of this table evenly distributed. TiDB also provides an interface in its HTTP API to simplify this operation. Refer to TiDB HTTP API for more details.

4.12.4.4.5 Region merge is slow

Similar to slow scheduling, the speed of region merge is most likely limited by the configurations of `merge-schedule-limit` and `region-schedule-limit`, or the region merge scheduler is competing with other schedulers. Specifically, the solutions are:

- If it is known from metrics that there are a large number of empty regions in the system, you can adjust `max-merge-region-size` and `max-merge-region-keys` to smaller values to speed up the merge. This is because the merge process involves replica migration, so the smaller the region to be merged, the faster the merge is. If the merge operators are already generated rapidly, to further speed up the process, you can set `patrol-region-interval` to `10ms`. This makes region scanning faster at the cost of more CPU consumption.
- A lot of tables have been created and then emptied (including truncated tables). These empty regions cannot be merged if the split table attribute is enabled. You can disable this attribute by adjusting the following parameters:
  - TiKV: Set `split-region-on-table` to `false`. You cannot modify the parameter dynamically.
  - PD: Set `namespace-classifier` to `"default"`. You cannot modify the parameter dynamically.

For v3.0.4 and v2.1.16 or earlier, the `approximate_keys` of regions are inaccurate in specific circumstances (most of which occur after dropping tables), which makes the number of keys break the constraints of `max-merge-region-keys`. To avoid this problem, you can adjust `max-merge-region-keys` to a larger value.

4.12.4.4.6 Troubleshoot TiKV node

If a TiKV node fails, PD defaults to setting the corresponding node to the **down** state after 30 minutes (customizable by configuration item `max-store-down-time`), and rebalancing replicas for regions involved.

Practically, if a node failure is considered unrecoverable, you can immediately take it offline. This makes PD replenish replicas soon in another node and reduces the risk of data loss. In contrast, if a node is considered recoverable, but the recovery cannot be done in 30 minutes, you can temporarily adjust `max-store-down-time` to a larger value to avoid unnecessary replenishment of the replicas and resources waste after the timeout.

4.12.5 Best Practices for Monitoring TiDB Using Grafana

When you deploy a TiDB cluster using TiDB Ansible, a set of Grafana + Prometheus monitoring platform is deployed simultaneously to collect and display metrics for various components and machines in the TiDB cluster. This document describes best practices for monitoring TiDB using Grafana. It aims to help you use metrics to analyze the status of the TiDB cluster and diagnose problems.

4.12.5.1 Monitoring architecture

Prometheus is a time series database with a multi-dimensional data model and a flexible query language. Grafana is an open source monitoring system for analyzing and visualizing metrics.

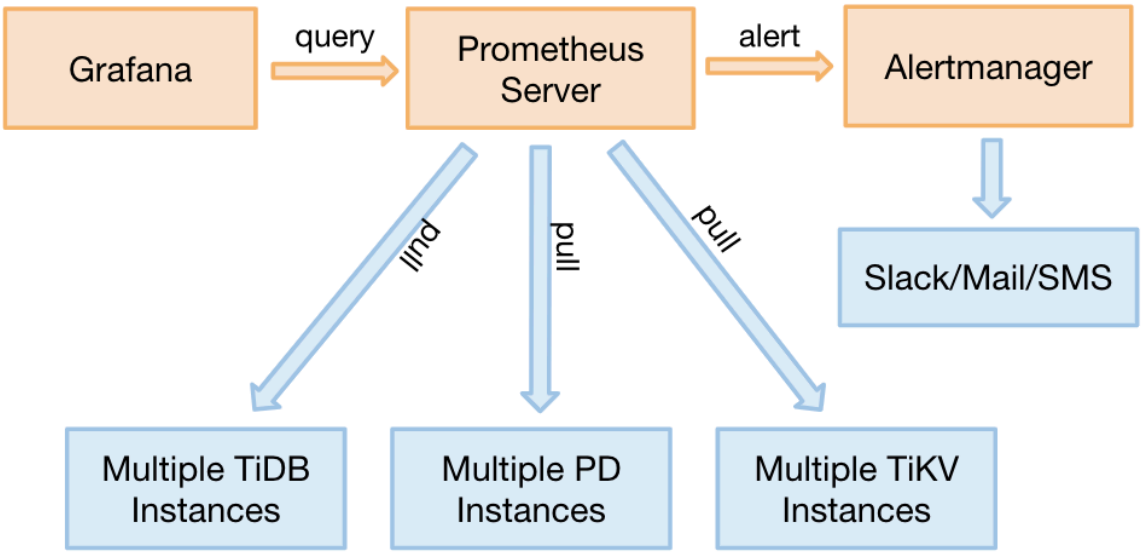

Figure 307: The monitoring architecture in the TiDB cluster

For TiDB 2.1.3 or later versions, TiDB monitoring supports the pull method. It is a good adjustment with the following benefits:

- There is no need to restart the entire TiDB cluster if you need to migrate Prometheus. Before adjustment, migrating Prometheus requires restarting the entire cluster because

the target address needs to be updated.
- You can deploy 2 separate sets of Grafana + Prometheus monitoring platforms (not highly available) to prevent a single point of monitoring. To do this, execute the deployment command of TiDB ansible twice with different IP addresses.
- The Pushgateway which might become a single point of failure is removed.

4.12.5.2 Source and display of monitoring data

The three core components of TiDB (TiDB server, TiKV server and PD server) obtain metrics through the HTTP interface. These metrics are collected from the program code, and the ports are as follows:

Component	Port
TiDB server	10080
TiKV server	20181
PD server	2379

Execute the following command to check the QPS of a SQL statement through the HTTP interface. Take the TiDB server as an example:

```
curl http://__tidb_ip__:10080/metrics |grep tidb_executor_statement_total
```

```
### Check the real-time QPS of different types of SQL statements. The
    ↪ numbers below are the cumulative values of counter type (scientific
    ↪ notation).
tidb_executor_statement_total{type="Delete"} 520197
tidb_executor_statement_total{type="Explain"} 1
tidb_executor_statement_total{type="Insert"} 7.20799402e+08
tidb_executor_statement_total{type="Select"} 2.64983586e+08
tidb_executor_statement_total{type="Set"} 2.399075e+06
tidb_executor_statement_total{type="Show"} 500531
tidb_executor_statement_total{type="Use"} 466016
```

The data above is stored in Prometheus and displayed on Grafana. Right-click the panel and then click the **Edit** button (or directly press the E key) shown in the following figure:

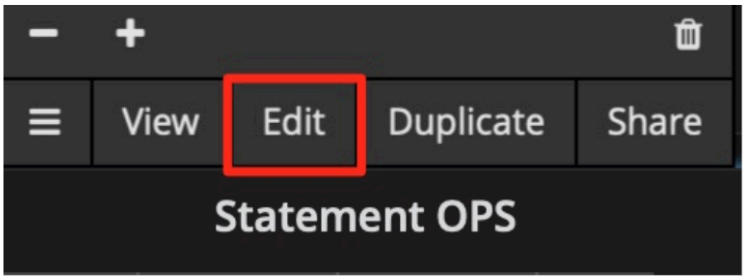

Figure 308: The Edit entry for the Metrics tab

After clicking the **Edit** button, you can see the query expression with the `tidb_executor_statement_t` ↪ metric name on the Metrics tab. The meanings of some items on the panel are as follows:

- `rate[1m]`: The growth rate in one minute. It can only be used for the data of counter type.
- `sum`: The sum of values.
- `by type`: The summed data is grouped by type in the original metric value.
- `Legend format`: The format of the metric name.
- `Resolution`: The step width defaults to 15 seconds. Resolution means whether to generate one data point for multiple pixels.

The query expression on the **Metrics** tab is as follows:

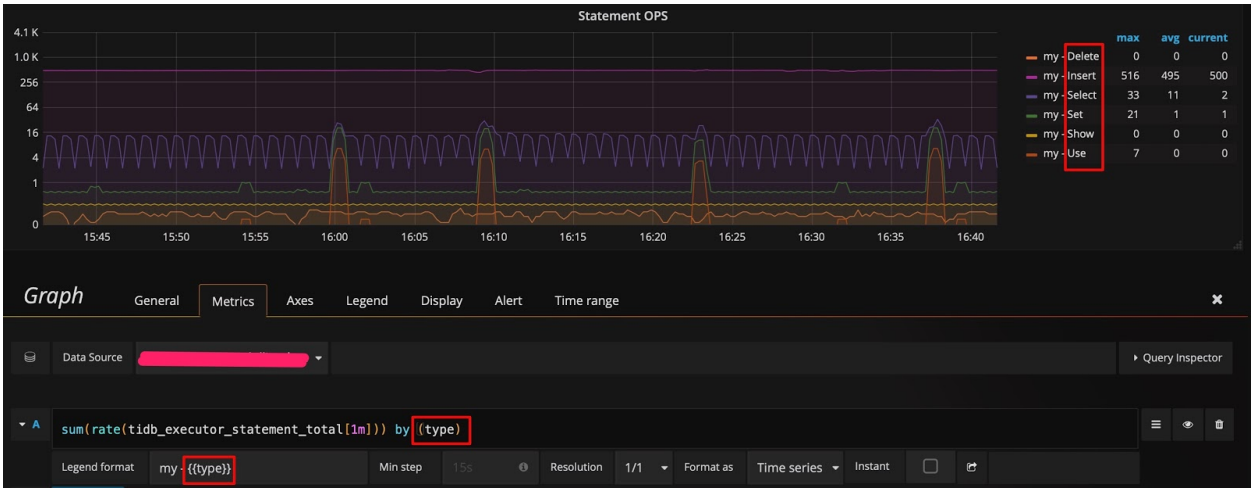

Figure 309: The query expression on the Metrics tab

Prometheus supports many query expressions and functions. For more details, refer to Prometheus official website.

4.12.5.3 Grafana tips

This section introduces seven tips for efficiently using Grafana to monitor and analyze the metrics of TiDB.

4.12.5.3.1 Tip 1: Check all dimensions and edit the query expression

In the example shown in the source and display of monitoring data section, the data is grouped by type. If you want to know whether you can group by other dimensions and quickly check which dimensions are available, you can use the following method: **Only keep the metric name on the query expression, no calculation, and leave the `Legend format` field blank**. In this way, the original metrics are displayed. For example, the following figure shows that there are three dimensions (`instance`, `job` and `type`):

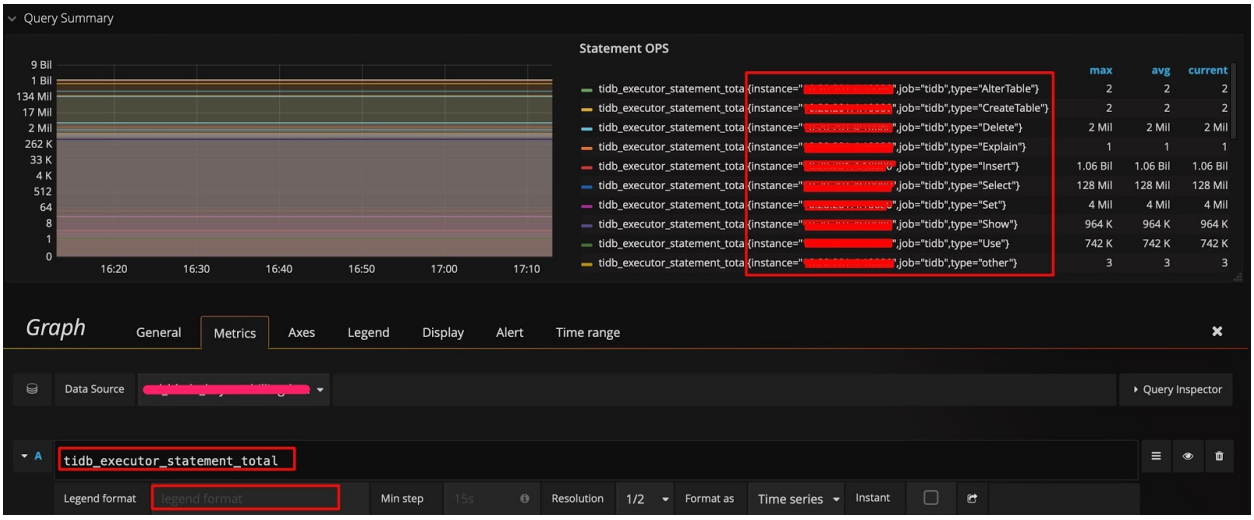

Figure 310: Edit query expression and check all dimensions

Then you can modify the query expression by adding the `instance` dimension after `type`, and adding `{{instance}}` to the `Legend format` field. In this way, you can check the QPS of different types of SQL statements that are executed on each TiDB server:

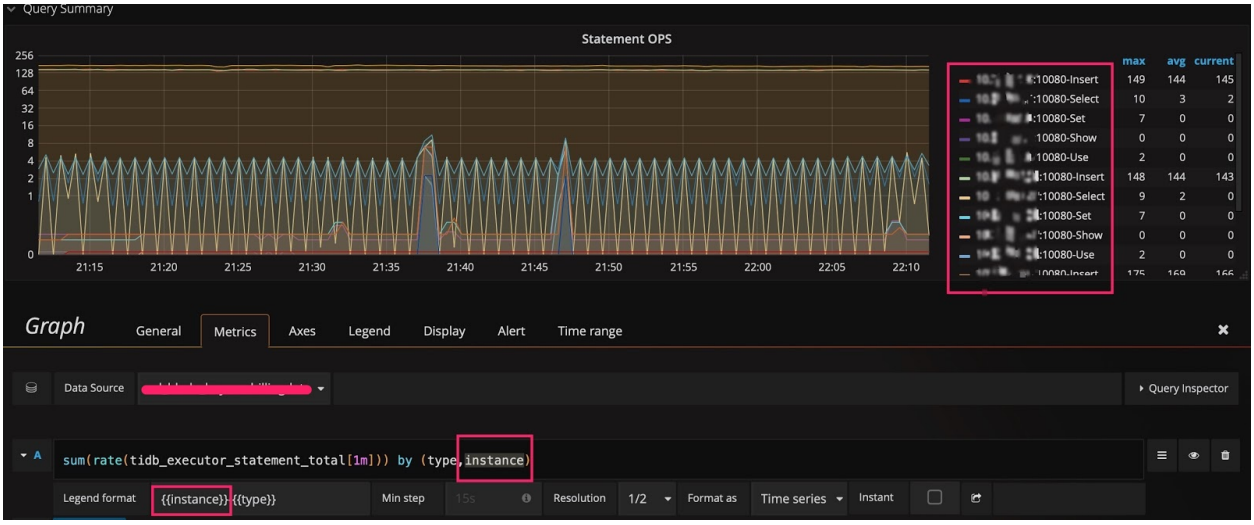

Figure 311: Add an instance dimension to the query expression

4.12.5.3.2 Tip 2: Switch the scale of the Y-axis

Take Query Duration as an example, the Y-axis defaults to be on a binary logarithmic scale (log2n), which narrows the gap in display. To amplify changes, you can switch it to a linear scale. Comparing the following two figures, you can easily notice the difference in display, and locate the time when an SQL statement runs slowly.

Of course, a linear scale is not suitable for all situations. For example, if you observe the performance trend for the duration of a month, there might be noises with a linear scale, making it hard to observe.

The Y-axis uses a binary logarithmic scale by default:

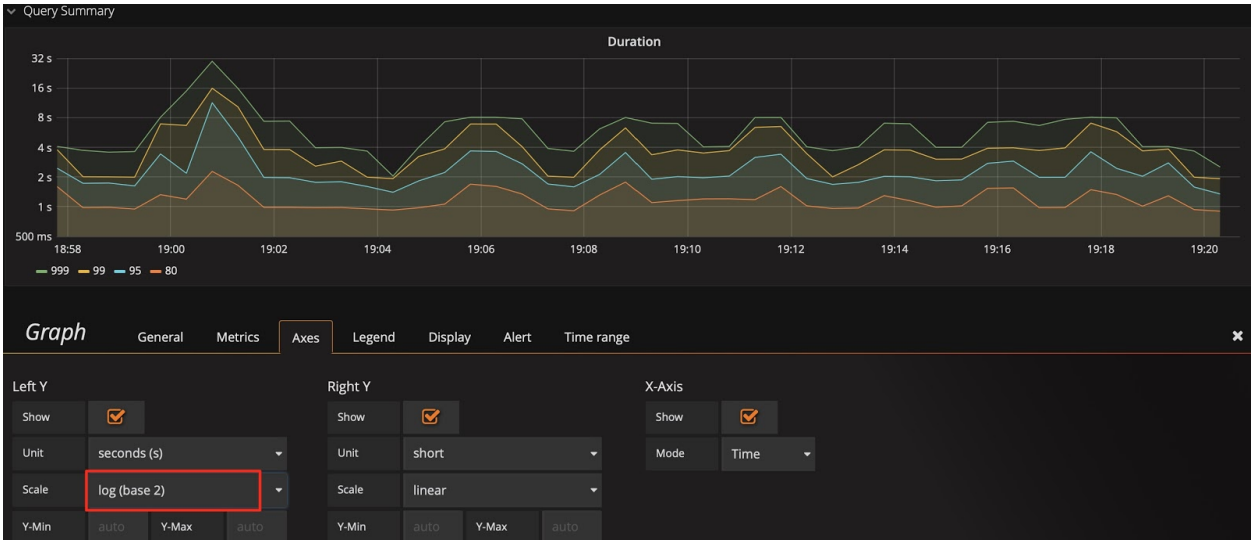

Figure 312: The Y-axis uses a binary logarithmic scale

Switch the Y-axis to a linear scale:

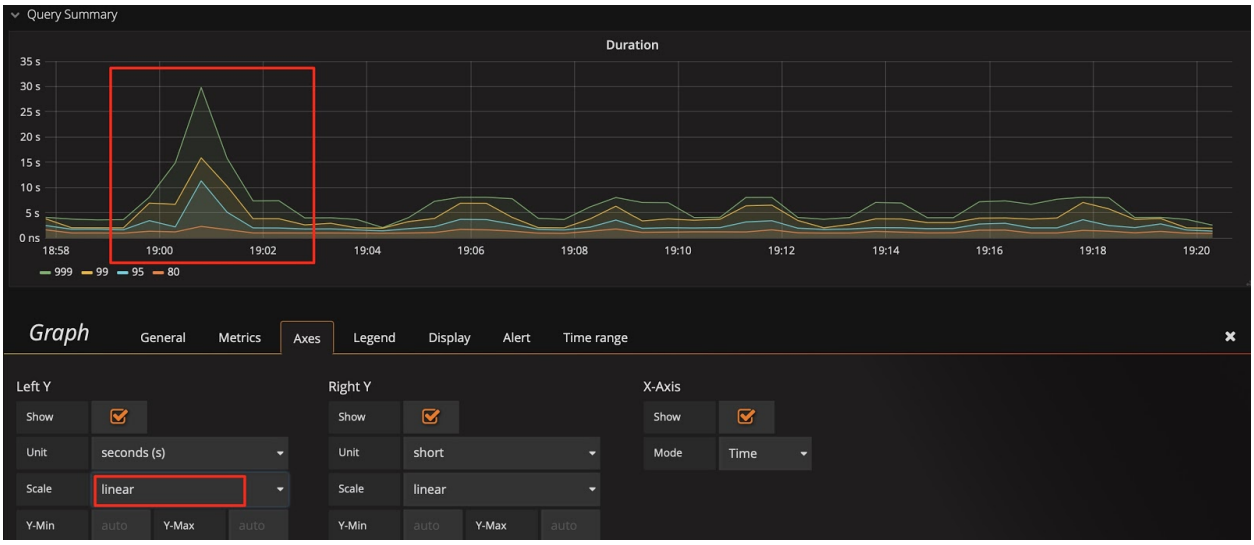

Figure 313: Switch to a linear scale

Tip:

Combining tip 2 with tip 1, you can find a `sql_type` dimension to help you immediately analyze whether the `SELECT` statement or the `UPDATE` statement is slow; you can even locate the instance with slow SQL statements.

4.12.5.3.3 Tip 3: Modify the baseline of the Y-axis to amplify changes

You might still cannot see the trend after switching to the linear scale. For example, in the following figure, you want to observe the real-time change of `Store size` after scaling the cluster, but due to the large baseline, small changes are not visible. In this situation, you can modify the baseline of the Y-axis from `0` to `auto` to zoom in the upper part. Check the two figures below, you can see data migration begins.

The baseline defaults to `0`:

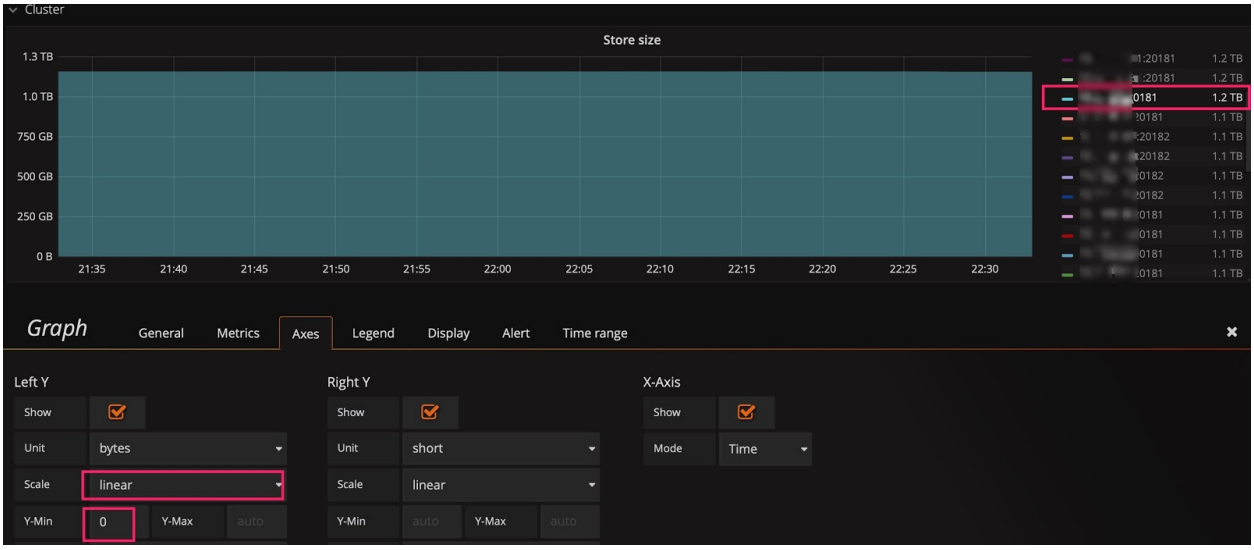

Figure 314: Baseline defaults to 0

Change the baseline to `auto`:

Figure 315: Change the baseline to auto

4.12.5.3.4 Tip 4: Use Shared crosshair or Tooltip

In the **Settings** panel, there is a **Graph Tooltip** panel option which defaults to **Default**.

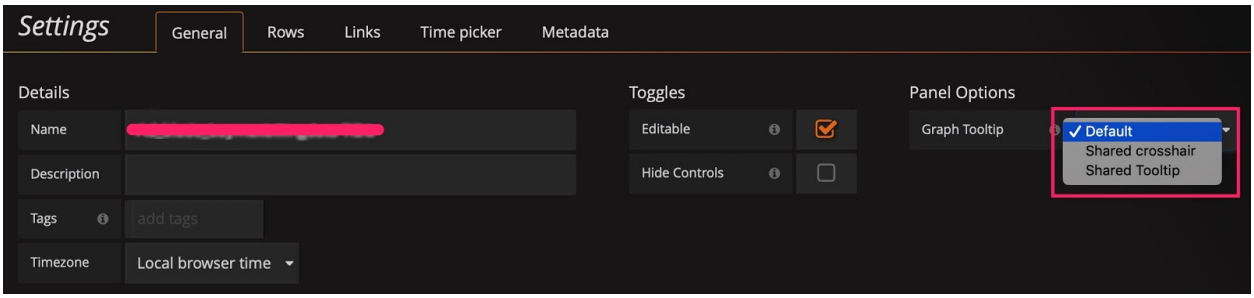

Figure 316: Graphic presentation tools

You can use **Shared crosshair** and **Shared Tooltip** respectively to test the effect as shown in the following figures. Then, the scales are displayed in linkage, which is convenient to confirm the correlation of two metrics when diagnosing problems.

Set the graphic presentation tool to **Shared crosshair**:

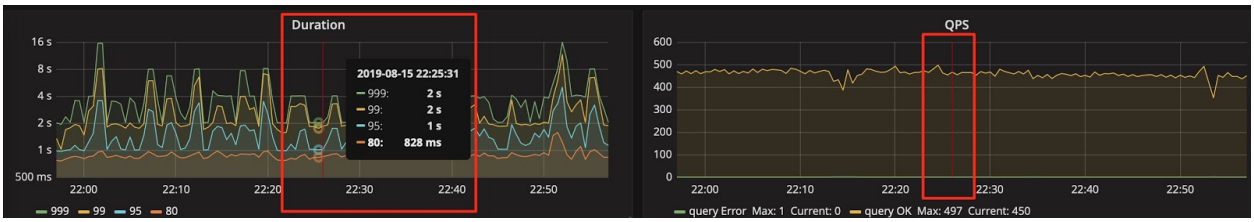

Figure 317: Set the graphical presentation tool to Shared crosshair

Set the graphical presentation tool to **Shared Tooltip**:

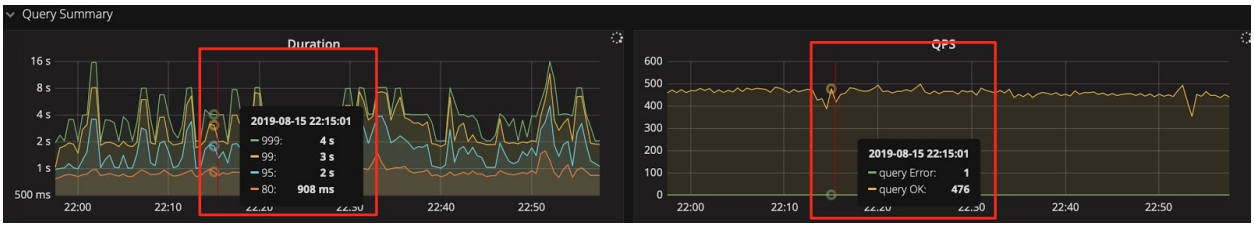

Figure 318: Set the graphic presentation tool to Shared Tooltip

4.12.5.3.5 Tip 5: Enter `IP address:port number` to check the metrics in history

PD's dashboard only shows the metrics of the current leader. If you want to check the status of a PD leader in history and it no longer exists in the drop-down list of the `instance` field, you can manually enter `IP address:2379` to check the data of the leader.

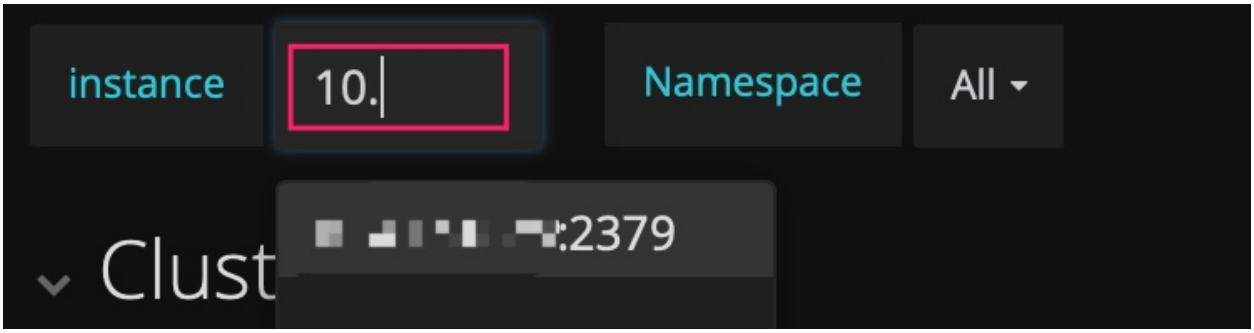

Figure 319: Check the metrics in history

4.12.5.3.6 Tip 6: Use the `Avg` function

Generally, only `Max` and `Current` functions are available in the legend by default. When the metrics fluctuate greatly, you can add other summary functions such as the `Avg` function to the legend to check the overall trend for the duration of time.

Add summary functions such as the `Avg` function:

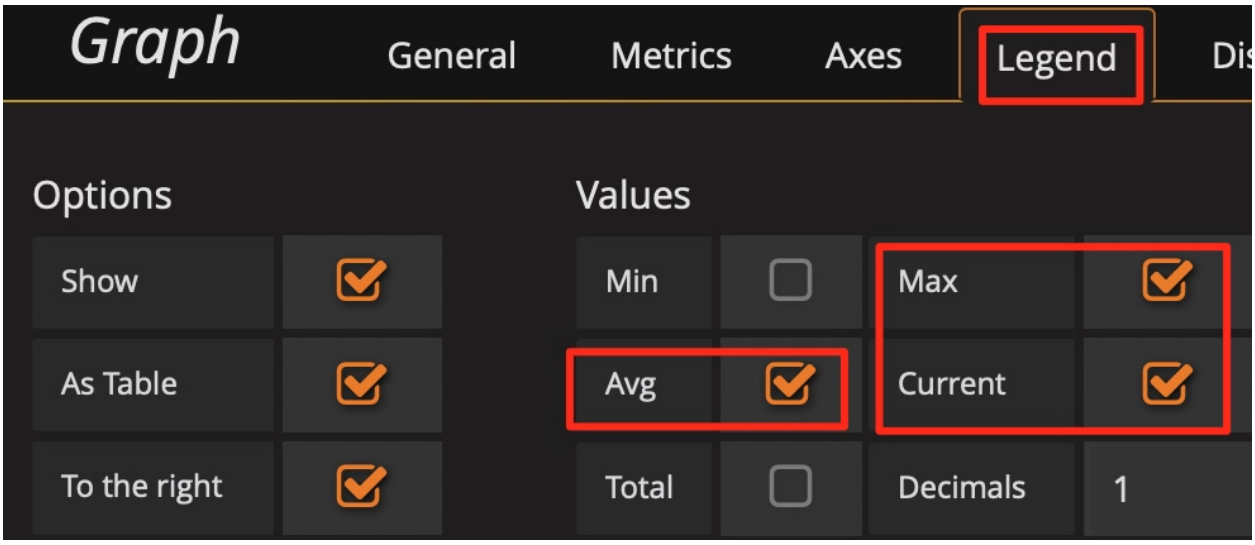

Figure 320: Add summary functions such as Avg

Then check the overall trend:

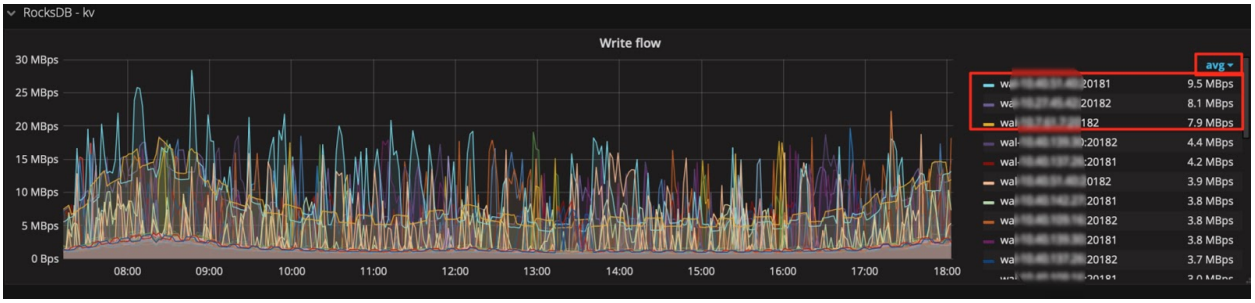

Figure 321: Add Avg function to check the overall trend

4.12.5.3.7 Tip 7: Use the API of Prometheus to obtain the result of query expressions

Grafana obtains data through the API of Prometheus and you can use this API to obtain information as well. In addition, it also has the following usages:

- Automatically obtains information such as the cluster size and status.
- Makes minor changes to the expression to provide information for the report, such as counting the total amount of QPS per day, the peak value of QPS per day, and the response time per day.
- Performs regular health inspection on the important metrics.

The API of Prometheus is shown as follows:

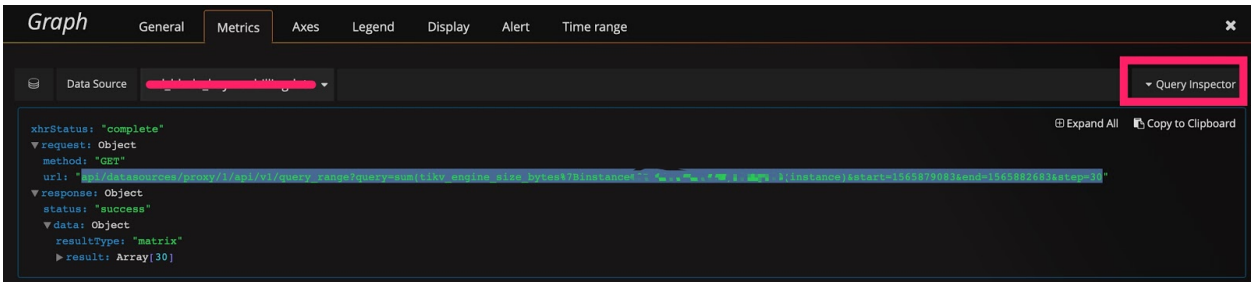

Figure 322: The API of Prometheus

```
curl -u user:pass 'http://__grafana_ip__:3000/api/datasources/proxy/1/api/v1
    ↪ /query_range?query=sum(tikv_engine_size_bytes%7
    ↪ Binstancexxxxxxxxx20181%22%7D)%20by%20(instance)&start=1565879269&end
    ↪ =1565882869&step=30' |python -m json.tool
```

```
{
    "data": {
        "result": [
            {
                "metric": {
                    "instance": "xxxxxxxxxx:20181"
                },
                "values": [
                    [
                        1565879269,
                        "1006046235280"
                    ],
                    [
                        1565879299,
                        "1006057877794"
                    ],
                    [
                        1565879329,
                        "1006021550039"
                    ],
                    [
                        1565879359,
                        "1006021550039"
                    ],
                    [
                        1565882869,
```

```
                    "1006132630123"
                ]
            ]
        }
    ],
    "resultType": "matrix"
},
"status": "success"
}
```

4.12.5.4 Summary

The Grafana + Prometheus monitoring platform is a very powerful tool. Making good use of it can improve efficiency, saving you a lot of time on analyzing the status of the TiDB cluster. More importantly, it can help you diagnose problems. This tool is very useful in the operation and maintenance of TiDB clusters, especially when there is a large amount of data.

4.12.6 Best Practices for TiKV Performance Tuning with Massive Regions

In TiDB, data is split into Regions, each storing data for a specific key range. These Regions are distributed among multiple TiKV instances. As data is written into a cluster, millions of or even tens of millions of Regions are created. Too many Regions on a single TiKV instance can bring a heavy burden to the cluster and affect its performance.

This document introduces the workflow of Raftstore (a core module of TiKV), explains why a massive amount of Regions affect the performance, and offers methods for tuning TiKV performance.

4.12.6.1 Raftstore workflow

A TiKV instance has multiple Regions on it. The Raftstore module drives the Raft state machine to process Region messages. These messages include processing read or write requests on Regions, persisting or replicating Raft logs, and processing Raft heartbeats. However, an increasing number of Regions can affect performance of the whole cluster. To understand this, it is necessary to learn the workflow of Raftstore shown as follows:

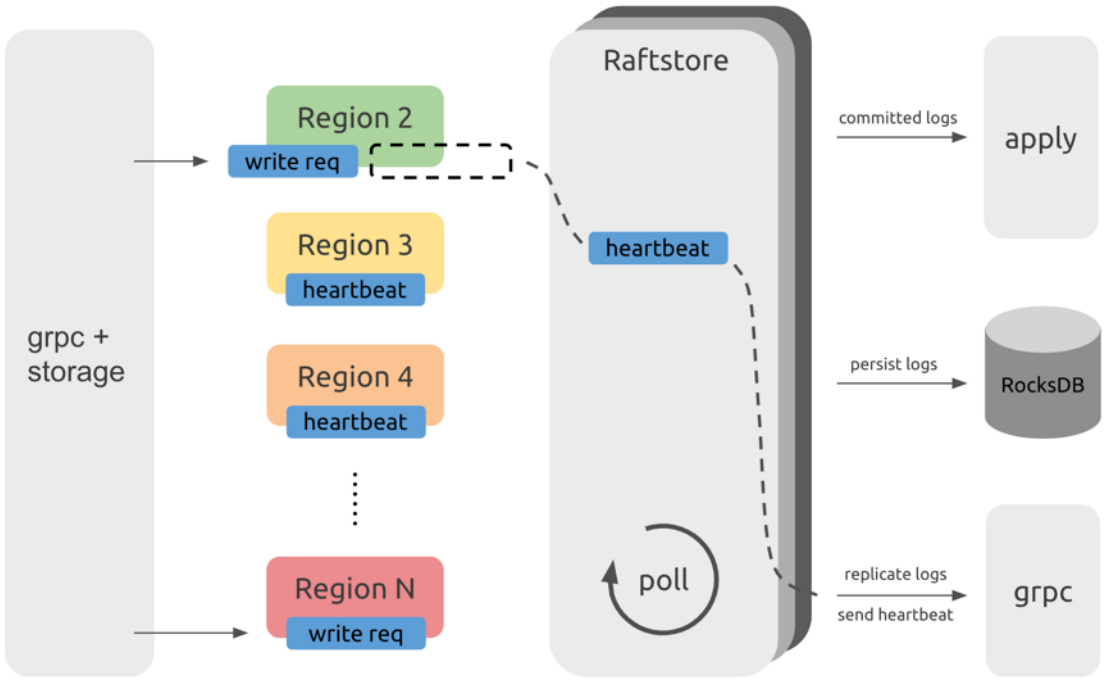

Figure 323: Raftstore Workflow

> **Note:**
>
> This diagram only illustrates the workflow of Raftstore and does not represent the actual code structure.

From the above diagram, you can see that requests from the TiDB servers, after passing through the gRPC and storage modules, become read and write messages of KV (key-value), and are sent to the corresponding Regions. These messages are not immediately processed but are temporarily stored. Raftstore polls to check whether each Region has messages to process. If a Region has messages to process, Raftstore drives the Raft state machine of this Region to process these messages and perform subsequent operations according to the state changes of these messages. For example, when write requests come in, the Raft state machine stores logs into disk and sends logs to other Region replicas; when the heartbeat interval is reached, the Raft state machine sends heartbeat information to other Region replicas.

4.12.6.2 Performance problem

From the Raftstore workflow diagram, messages in each Region are processed one by one. When a large number of Regions exist, it takes Raftstore some time to process the heartbeats of these Regions, which can cause some delay. As a result, some read and write requests are not processed in time. If read and write pressure is high, the CPU usage of the

Raftstore thread might easily become the bottleneck, which further increases the delay and affects the performance.

Generally, if the CPU usage of the loaded Raftstore reaches 85% or higher, Raftstore goes into a busy state and becomes the bottleneck. At the same time, `propose wait duration` can be as high as hundreds of milliseconds.

Note:

- For the CPU usage of Raftstore as mentioned above, Raftstore is single-threaded. If Raftstore is multi-threaded, you can increase the CPU usage threshold (85%) proportionally.
- Because I/O operations exist in the Raftstore thread, CPU usage cannot reach 100%.

4.12.6.2.1 Performance monitoring

You can check the following monitoring metrics in Grafana's **TiKV Dashboard**:

- `Raft store CPU` in the **Thread-CPU** panel

  Reference value: lower than `raftstore.store-pool-size * 85%`. TiDB v2.1 does not have the `raftstore.store-pool-size` configuration item, so you can take this item's value as `1` in v2.1 versions.

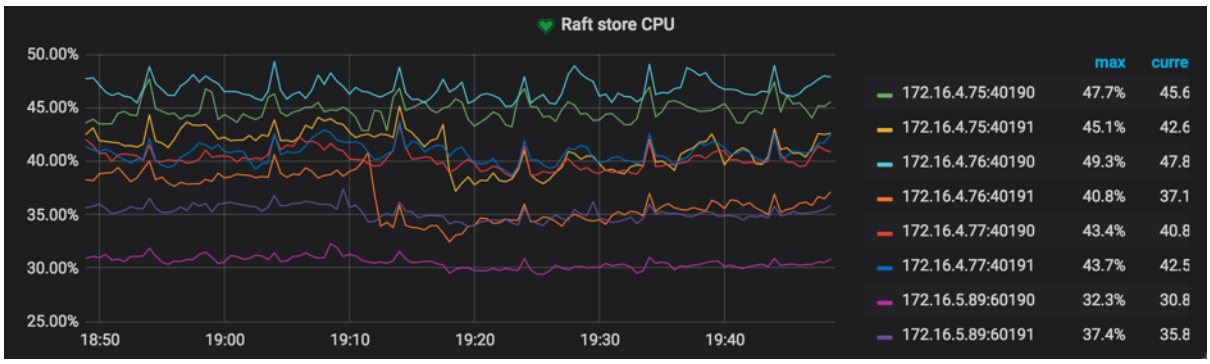

Figure 324: Check Raftstore CPU

- `Propose wait duration` in the **Raft Propose** panel

  `Propose wait duration` is the delay between the time a request is sent to Raftstore and the time Raftstore actually starts processing the request. Long delay means that Raftstore is busy, or that processing the append log is time-consuming, making Raftstore unable to process the request in time.

Reference value: lower than 50~100 ms according to the cluster size

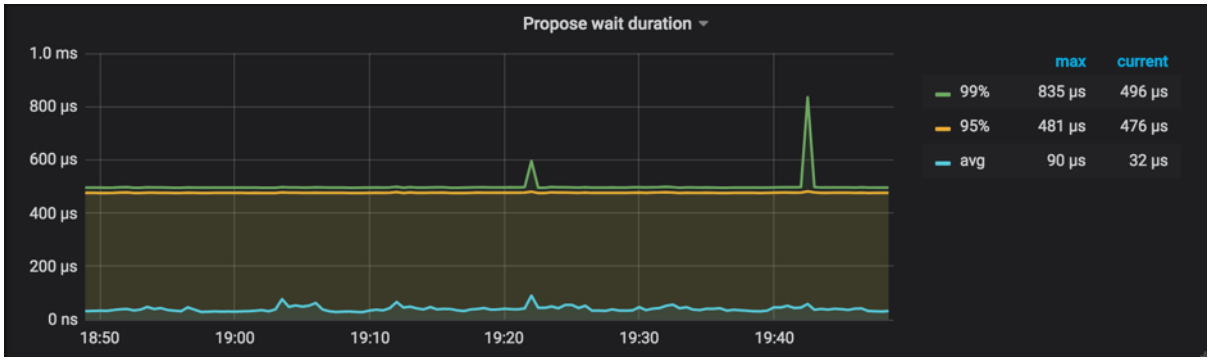

Figure 325: Check Propose wait duration

4.12.6.3 Performance tuning methods

After finding out the cause of a performance problem, try to solve it from the following two aspects:

- Reduce the number of Regions on a single TiKV instance
- Reduce the number of messages for a single Region

4.12.6.3.1 Method 1: Increase Raftstore concurrency

Raftstore in TiDB v3.0 has been upgraded to a multi-threaded module, which greatly reduces the possibility that a Raftstore thread becomes the bottleneck.

By default, `raftstore.store-pool-size` is configured to `2` in TiKV. If a bottleneck occurs in Raftstore, you can properly increase the value of this configuration item according to the actual situation. But to avoid introducing unnecessary thread switching overhead, it is recommended that you do not set this value too high.

4.12.6.3.2 Method 2: Enable Hibernate Region

In the actual situation, read and write requests are not evenly distributed on every Region. Instead, they are concentrated on a few Regions. Then you can minimize the number of messages between the Raft leader and the followers for the temporarily idle Regions, which is the feature of Hibernate Region. In this feature, Raftstore does sent tick messages to the Raft state machines of idle Regions if not necessary. Then these Raft state machines will not be triggered to generate heartbeat messages, which can greatly reduce the workload of Raftstore.

Up to TiDB v3.0.9, Hibernate Region is still an experimental feature, which is enabled by default in TiKV master. You can enable this feature according to your needs. For the configuration of Hibernate Region, refer to Configure Hibernate Region.

4.12.6.3.3 Method 3: Enable `Region Merge`

> **Note:**
>
> `Region Merge` is enabled in TiDB v3.0 by default.

You can also reduce the number of Regions by enabling `Region Merge`. Contrary to `Region Split`, `Region Merge` is the process of merging adjacent small Regions through scheduling. After dropping data or executing the `Drop Table` or `Truncate Table` statement, you can merge small Regions or even empty Regions to reduce resource consumption.

Enable `Region Merge` by configuring the following parameters:

```
>> pd-ctl config set max-merge-region-size 20
>> pd-ctl config set max-merge-region-keys 200000
>> pd-ctl config set merge-schedule-limit 8
```

Refer to Region Merge and the following three configuration parameters in the PD configuration file for more details:

- `max-merge-region-size`
- `max-merge-region-keys`
- `merge-schedule-limit`

The default configuration of the `Region Merge` parameters is rather conservative. You can speed up the `Region Merge` process by referring to the method provided in PD Scheduling Best Practices.

4.12.6.3.4 Method 4: Increase the number of TiKV instances

If I/O resources and CPU resources are sufficient, you can deploy multiple TiKV instances on a single machine to reduce the number of Regions on a single TiKV instance; or you can increase the number of machines in the TiKV cluster.

4.12.6.3.5 Method 5: Adjust `raft-base-tick-interval`

In addition to reducing the number of Regions, you can also reduce pressure on Raftstore by reducing the number of messages for each Region within a unit of time. For example, you can properly increase the value of the `raft-base-tick-interval` configuration item:

```
[raftstore]
raft-base-tick-interval = "2s"
```

In the above configuration, `raft-base-tick-interval` is the time interval at which Raftstore drives the Raft state machine of each Region, which means at this time interval, Raftstore sends a tick message to the Raft state machine. Increasing this interval can effectively reduce the number of messages from Raftstore.

Note that this interval between tick messages also determines the intervals between `election timeout` and `heartbeat`. See the following example:

```
raft-election-timeout = raft-base-tick-interval * raft-election-timeout-
    ↪ ticks
raft-heartbeat-interval = raft-base-tick-interval * raft-heartbeat-ticks
```

If Region followers have not received the heartbeat from the leader within the `raft ↪ -election-timeout` interval, these followers determine that the leader has failed and start a new election. `raft-heartbeat-interval` is the interval at which a leader sends a heartbeat to followers. Therefore, increasing the value of `raft-base-tick-interval` can reduce the number of network messages sent from Raft state machines but also makes it longer for Raft state machines to detect the leader failure.

4.12.6.4 Other problems and solutions

This section describes some other problems and solutions.

4.12.6.4.1 Switching PD Leader is slow

PD needs to persist Region Meta information on etcd to ensure that PD can quickly resume to provide Region routing services after switching the PD Leader node. As the number of Regions increases, the performance problem of etcd appears, making it slower for PD to get Region Meta information from etcd when PD is switching the Leader. With millions of Regions, it might take more than ten seconds or even tens of seconds to get the meta information from etcd.

To address this problem, `use-region-storage` is enabled by default in PD in TiDB v3.0. With this feature enabled, PD stores Region Meta information on local LevelDB and synchronizes the information among PD nodes through other mechanisms.

4.12.6.4.2 PD routing information is not updated in time

In TiKV, pd-worker regularly reports Region Meta information to PD. When TiKV is restarted or switches the Region leader, PD needs to recalculate Region's `approximate ↪ size / keys` through statistics. Therefore, with a large number of Regions, the single-threaded pd-worker might become the bottleneck, causing tasks to be piled up and not processed in time. In this situation, PD cannot obtain certain Region Meta information in time so that the routing information is not updated in time. This problem does not affect the actual reads and writes, but might cause inaccurate PD scheduling and require several round trips when TiDB updates Region cache.

You can check **Worker pending tasks** under **Task** in the **TiKV Grafana** panel to determine whether pd-worker has tasks piled up. Generally, `pending tasks` should be kept at a relatively low value.

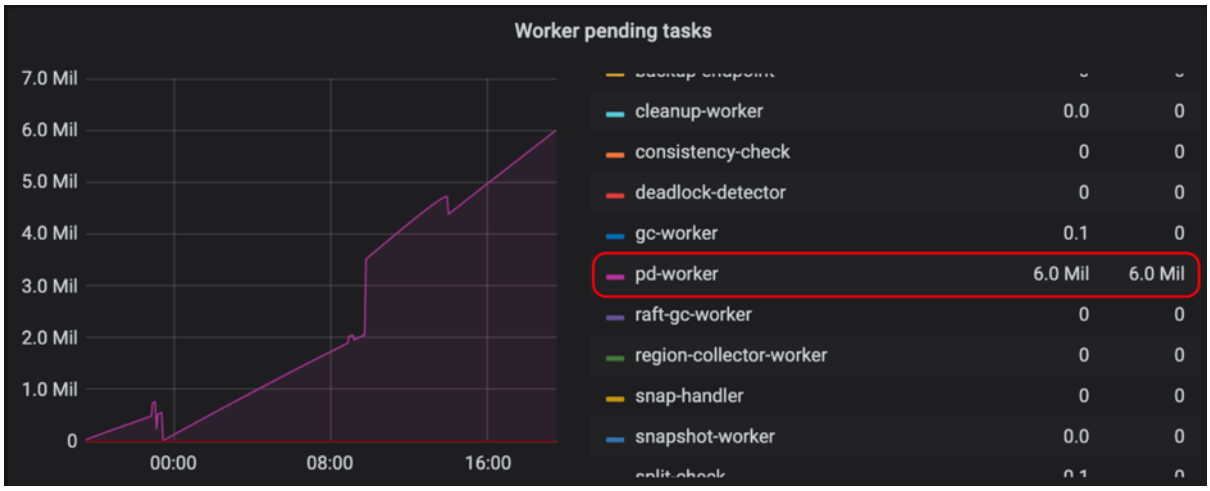

Figure 326: Check pd-worker

Currently, pd-worker is optimized for better efficiency in #5620 on TiKV master, which is applied since v3.0.5. If you encounter a similar problem, it is recommended to upgrade to v3.0.5 or later versions.

4.12.6.4.3 Prometheus is slow to query metrics

In a large-scale cluster, as the number of TiKV instances increases, Prometheus has greater pressure to query metrics, making it slower for Grafana to display these metrics. To ease this problem, metrics pre-calculation is configured in v3.0.

4.13 TiSpark User Guide

TiSpark is a thin layer built for running Apache Spark on top of TiDB/TiKV to answer the complex OLAP queries. It takes advantages of both the Spark platform and the distributed TiKV cluster and seamlessly glues to TiDB, the distributed OLTP database, to provide a Hybrid Transactional/Analytical Processing (HTAP) solution to serve as a one-stop solution for both online transactions and analysis.

TiSpark depends on the TiKV cluster and the PD cluster. You also need to set up a Spark cluster. This document provides a brief introduction to how to setup and use TiSpark. It requires some basic knowledge of Apache Spark. For more information, see Spark website.

4.13.1 Overview

TiSpark is an OLAP solution that runs Spark SQL directly on TiKV, the distributed storage engine.

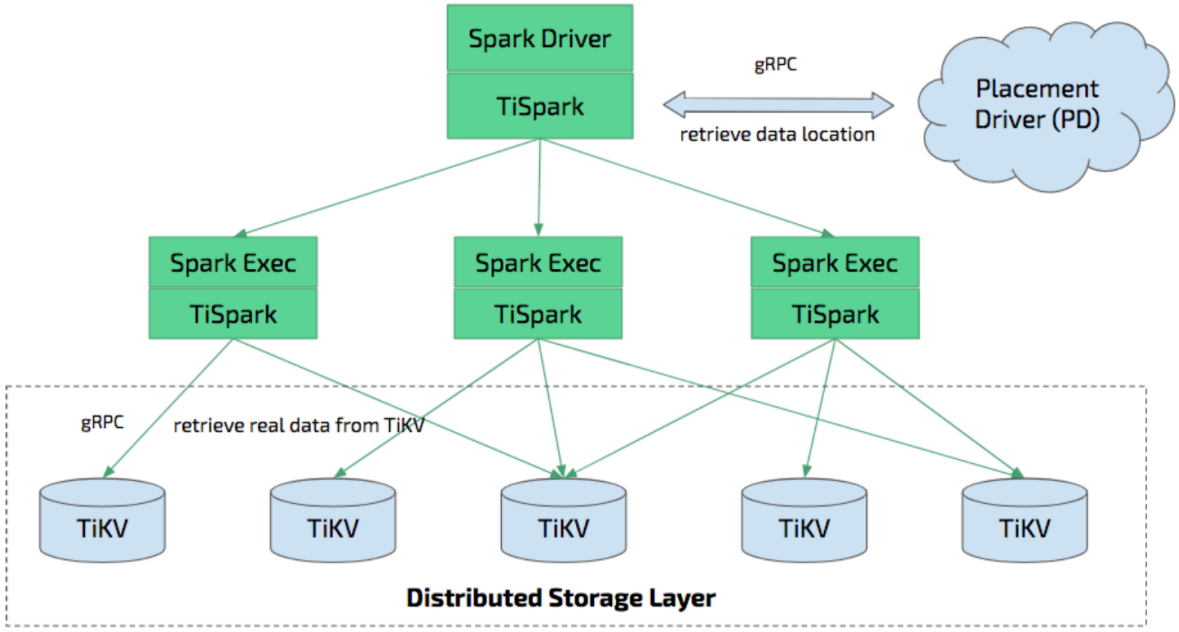

Figure 327: TiSpark architecture

- TiSpark integrates with Spark Catalyst Engine deeply. It provides precise control of the computing, which allows Spark read data from TiKV efficiently. It also supports index seek, which improves the performance of the point query execution significantly.
- It utilizes several strategies to push down the computing to reduce the size of dataset handling by Spark SQL, which accelerates the query execution. It also uses the TiDB built-in statistical information for the query plan optimization.
- From the data integration point of view, TiSpark and TiDB serve as a solution for running both transaction and analysis directly on the same platform without building and maintaining any ETLs. It simplifies the system architecture and reduces the cost of maintenance.
- You can deploy and utilize tools from the Spark ecosystem for further data processing and manipulation on TiDB. For example, using TiSpark for data analysis and ETL; retrieving data from TiKV as a machine learning data source; generating reports from the scheduling system and so on.
- Also, TiSpark supports distributed writes to TiKV. Compared to using Spark combined with JDBC to write to TiDB, distributed writes to TiKV can implement transactions (either all data are written successfully or all writes fail), and the writes are faster.

4.13.2 Environment setup

- The TiSpark 2.x supports Spark 2.3.x and Spark 2.4.x. If you want to use Spark 2.1.x, use TiSpark 1.x instead.

- TiSpark requires JDK 1.8+ and Scala 2.11 (Spark2.0 + default Scala version).
- TiSpark runs in any Spark mode such as YARN, Mesos, and Standalone.

4.13.3 Recommended configuration

This section describes the configuration of independent deployment of TiKV and TiSpark, independent deployment of Spark and TiSpark, and hybrid deployment of TiKV and TiSpark.

4.13.3.1 Configuration of independent deployment of TiKV and TiSpark

For independent deployment of TiKV and TiSpark, it is recommended to refer to the following recommendations:

- Hardware configuration
  - For general purposes, refer to the TiDB and TiKV hardware configuration recommendations.
  - If the usage is more focused on the analysis scenarios, you can increase the memory of the TiKV nodes to at least 64G.

4.13.3.2 Configuration of independent deployment of Spark and TiSpark

See the Spark official website for the detail hardware recommendations.

The following is a short overview of TiSpark configuration.

It is recommended to allocate 32G memory for Spark, and reserve at least 25% of the memory for the operating system and buffer cache.

It is recommended to provision at least 8 to 16 cores on per machine for Spark. Initially, you can assign all the CPU cores to Spark.

See the official configuration on the Spark website. The following is an example based on the `spark-env.sh` configuration:

```
SPARK_EXECUTOR_CORES: 5
SPARK_EXECUTOR_MEMORY: 10g
SPARK_WORKER_CORES: 5
SPARK_WORKER_MEMORY: 10g
```

In the `spark-defaults.conf` file, add the following lines:

```
spark.tispark.pd.addresses $your_pd_servers
spark.sql.extensions org.apache.spark.sql.TiExtensions
```

Add the following configuration in the `CDH` spark version:

```
spark.tispark.pd.addresses=$your_pd_servers
spark.sql.extensions=org.apache.spark.sql.TiExtensions
```

`your_pd_servers` are comma-separated PD addresses, with each in the format of `$your_pd_address:$port`.

For example, when you have multiple PD servers on `10.16.20.1,10.16.20.2,10.16.20.3` with the port 2379, put it as `10.16.20.1:2379,10.16.20.2:2379,10.16.20.3:2379`.

4.13.3.3 Configuration of hybrid deployment of TiKV and TiSpark

For the hybrid deployment of TiKV and TiSpark, add TiSpark required resources to the TiKV reserved resources, and allocate 25% of the memory for the system.

4.13.4 Deploy the TiSpark cluster

Download TiSpark's jar package in the TiSpark releases page. Download your desired version of jar package and copy the content to the appropriate folder.

4.13.4.1 Deploy TiSpark on the existing Spark cluster

Running TiSpark on an existing Spark cluster does not require a reboot of the cluster. You can use Spark's `--jars` parameter to introduce TiSpark as a dependency:

```
spark-shell --jars $TISPARK_FOLDER/tispark-${name_with_version}.jar
```

4.13.4.2 Deploy TiSpark without the Spark cluster

If you do not have a Spark cluster, we recommend using the standalone mode. To use the Spark Standalone model, you can simply place a compiled version of Spark on each node of the cluster. If you encounter problems, see its official website. And you are welcome to file an issue on our GitHub.

If you are using TiDB Ansible to deploy a TiDB cluster, you can also use TiDB Ansible to deploy a Spark standalone cluster, and TiSpark is also deployed at the same time.

4.13.4.2.1 Download and install

You can download Apache Spark

For the Standalone mode without Hadoop support, use Spark **2.3.x** and any version of Pre-build with Apache Hadoop 2.x with Hadoop dependencies. If you need to use the Hadoop cluster, choose the corresponding Hadoop version. You can also choose to build from the source code to match the previous version of the official Hadoop 2.x.

Suppose you already have a Spark binaries, and the current PATH is `SPARKPATH`, you can copy the TiSpark jar package to the `${SPARKPATH}/jars` directory.

4.13.4.2.2 Start a Master node

Execute the following command on the selected Spark Master node:

```
cd $SPARKPATH

./sbin/start-master.sh
```

After the above step is completed, a log file will be printed on the screen. Check the log file to confirm whether the Spark-Master is started successfully. You can open the http://spark-master-hostname:8080 to view the cluster information (if you does not change the Spark-Master default port number). When you start Spark-Worker, you can also use this panel to confirm whether the Worker is joined to the cluster.

4.13.4.2.3 Start a Worker node

Similarly, you can start a Spark-Worker node with the following command:

```
./sbin/start-slave.sh spark://spark-master-hostname:7077
```

After the command returns, you can see if the Worker node is joined to the Spark cluster correctly from the panel as well. Repeat the above command at all Worker nodes. After all Workers are connected to the master, you have a Standalone mode Spark cluster.

4.13.4.2.4 Spark SQL shell and JDBC server

TiSpark supports Spark 2.3, so you can use Spark's ThriftServer and SparkSQL directly.

4.13.5 Demo

Assuming that you have successfully started the TiSpark cluster as described above, here's a quick introduction to how to use Spark SQL for OLAP analysis. Here we use a table named `lineitem` in the `tpch` database as an example.

Assuming that your PD node is located at `192.168.1.100`, port `2379`, add the following command to `$SPARK_HOME/conf/spark-defaults.conf`:

```
spark.tispark.pd.addresses 192.168.1.100:2379
spark.sql.extensions org.apache.spark.sql.TiExtensions
```

And then enter the following command in the Spark-Shell as in native Apache Spark:

```
spark.sql("use tpch")

spark.sql("select count(*)from lineitem").show
```

The result is:

```
+-------------+
| Count (1) |
+-------------+
| 600000000 |
+-------------+
```

Spark SQL Interactive shell remains the same:

```
spark-sql> use tpch;
Time taken: 0.015 seconds

spark-sql> select count(*) from lineitem;
2000
Time taken: 0.673 seconds, Fetched 1 row(s)
```

For JDBC connection with Thrift Server, you can try it with various JDBC supported tools including SQuirreLSQL and hive-beeline. For example, to use it with beeline:

```
./beeline
Beeline version 1.2.2 by Apache Hive
beeline> !connect jdbc:hive2://localhost:10000

1: jdbc:hive2://localhost:10000> use testdb;
+---------+--+
| Result |
+---------+--+
+---------+--+
No rows selected (0.013 seconds)

select count(*) from account;
+-----------+--+
| count(1) |
+-----------+--+
| 1000000 |
+-----------+--+
1 row selected (1.97 seconds)
```

4.13.6 Use TiSpark together with Hive

You can use TiSpark together with Hive.

Before starting Spark, you need to set the `HADOOP_CONF_DIR` environment variable to your Hadoop configuration folder and copy `hive-site.xml` to the `spark/conf` folder.

```
val tisparkDF = spark.sql("select * from tispark_table").toDF
```

```
tisparkDF.write.saveAsTable("hive_table") // save table to hive
spark.sql("select * from hive_table a, tispark_table b where a.col1 = b.col1
    ↪ ").show // join table across Hive and Tispark
```

4.13.7 Load Spark Dataframe into TiDB using JDBC

TiSpark does not provide a direct way of loading data into your TiDB cluster, but you can load data using JDBC like this:

```
import org.apache.spark.sql.execution.datasources.jdbc.JDBCOptions

val customer = spark.sql("select * from customer limit 100000")
// You might repartition the source to make it balance across nodes
// and increase the concurrency.
val df = customer.repartition(32)
df.write
.mode(saveMode = "append")
.format("jdbc")
.option("driver", "com.mysql.jdbc.Driver")
 // Replace the host and port with that of your own and be sure to use the
    ↪  rewrite batch
.option("url", "jdbc:mysql://127.0.0.1:4000/test?rewriteBatchedStatements=
    ↪ true")
.option("useSSL", "false")
// As tested, 150 is good practice
.option(JDBCOptions.JDBC_BATCH_INSERT_SIZE, 150)
.option("dbtable", s"cust_test_select") // database name and table name
    ↪ here
.option("isolationLevel", "NONE") // recommended to set isolationLevel to
    ↪ NONE if you have a large DF to load.
.option("user", "root") // TiDB user here
.save()
```

It is recommended to set `isolationLevel` to `NONE` to avoid large single transactions which might potentially lead to TiDB OOM.

4.13.8 Statistics information

TiSpark uses TiDB statistic information for the following items:

1. Determining which index to ues in your query plan with the estimated lowest cost.
2. Small table broadcasting, which enables efficient broadcast join.

If you would like TiSpark to use statistic information, first you need to make sure that concerning tables have already been analyzed. Read more about how to analyze tables.

Starting from TiSpark 2.0, statistics information is default to auto load.

Note that table statistics are cached in the memory of your Spark driver node, so you need to make sure that your memory size is large enough for your statistics information.

Currently, you can adjust these configurations in your `spark-defaults.conf` file.

Property name	Default	Description
spark.tispark.statistics.auto_load	true	Whether to load statistics information automatically during database mapping.

4.13.9 FAQ

Q: What are the pros/cons of independent deployment as opposed to a shared resource with an existing Spark / Hadoop cluster?

A: You can use the existing Spark cluster without a separate deployment, but if the existing cluster is busy, TiSpark will not be able to achieve the desired speed.

Q: Can I mix Spark with TiKV?

A: If TiDB and TiKV are overloaded and run critical online tasks, consider deploying TiSpark separately. You also need to consider using different NICs to ensure that OLTP's network resources are not compromised and affect online business. If the online business requirements are not high or the loading is not large enough, you can consider mixing TiSpark with TiKV deployment.

Q: What can I do if `warning: WARN ObjectStore:568 - Failed to get database` is returned when executing SQL statements using TiSpark?

A: You can ignore this warning. It occurs because Spark tries to load two nonexistent databases (`default` and `global_temp`) in its catalog. If you want to mute this warning, modify log4j by adding `log4j.logger.org.apache.hadoop.hive.metastore.ObjectStore=ERROR` to the `log4j` file in `tispark/conf`. You can add the parameter to the `log4j` file of the `config` under Spark. If the suffix is `template`, you can use the `mv` command to change it to `properties`.

Q: What can I do if `java.sql.BatchUpdateException: Data Truncated` is returned when executing SQL statements using TiSpark?

A: This error occurs because the length of the data written exceeds the length of the data type defined by the database. You can check the field length and adjust it accordingly.

Q: Does TiSpark read Hive metadata by default?

A: By default, TiSpark searches for the Hive database by reading the Hive metadata in hive-site. If the search task fails, it searches for the TiDB database instead, by reading the TiDB metadata.

If you do not need this default behavior, do not configure the Hive metadata in hive-site.

Q: What can I do if `Error: java.io.InvalidClassException: com.pingcap.tikv.region.TiRegion; local class incompatible: stream classdesc serialVersionUID ... ` is returned when TiSpark is executing a Spark task?

A: The error message shows a `serialVersionUID` conflict, which occurs because you have used `class` and `TiRegion` of different versions. Because `TiRegion` only exists in TiSpark, multiple versions of TiSpark packages might be used. To fix this error, you need to make sure the version of TiSpark dependency is consistent among all nodes in the cluster.

4.14 TiKV Overview

TiKV is a distributed and transactional key-value database, which provides transactional APIs with ACID compliance. With the implementation of the Raft consensus algorithm and consensus state stored in RocksDB, TiKV guarantees data consistency between multiple replicas and high availability. As the storage layer of the TiDB distributed database, TiKV provides the read and write service, and persist the written data from applications. It also stores the statistics data of the TiDB cluster.

4.14.1 Architecture Overview

TiKV implements the multi-raft-group replica mechanism based on the design of Google Spanner. A Region is a basic unit of the key-value data movement and refers to a data range in a Store. Each Region is replicated to multiple nodes. These multiple replicas form a Raft group. A replica of a Region is called a Peer. Typically there are 3 peers in a Region. One of them is the leader, which provides the read and write services. The PD component balances all the Regions automatically to guarantee that the read and write throughput is balanced among all the nodes in the TiKV cluster. With PD and carefully designed Raft

groups, TiKV excels in horizontal scalability and can easily scale to store more than 100 TBs of data.

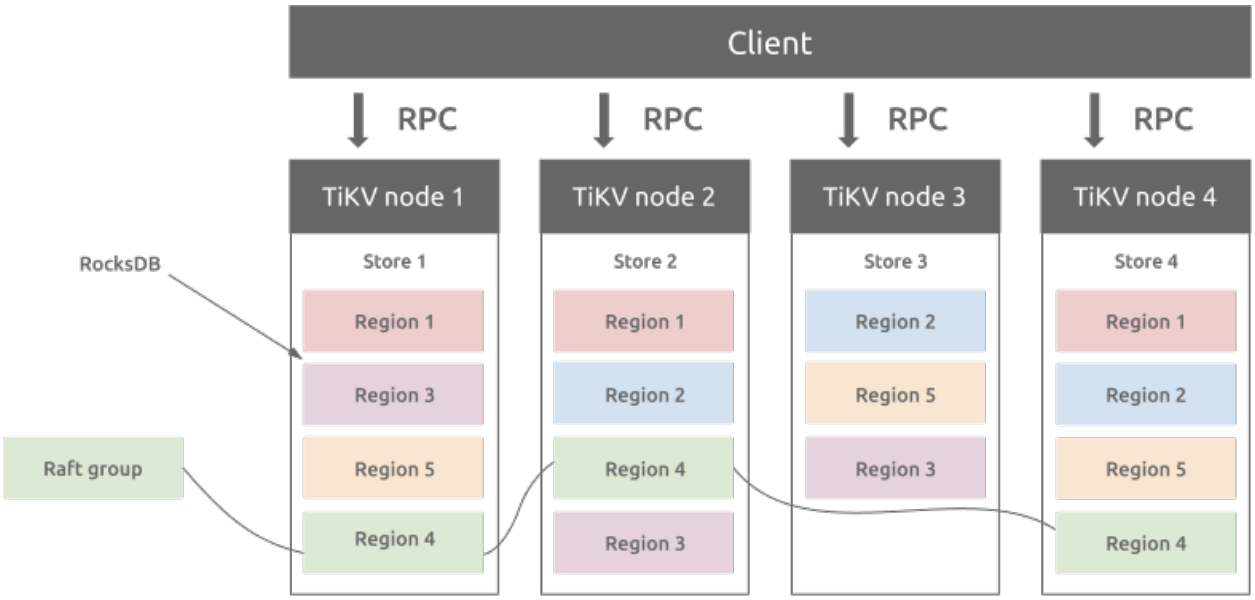

Figure 328: TiKV Architecture

4.14.1.1 Region and RocksDB

There is a RocksDB database within each Store and it stores data into the local disk. All the Region data are stored in the same RocksDB instance in each Store. All the logs used for the Raft consensus algorithm is stored in another RocksDB instance in each Store. This is because the performance of sequential I/O is better than random I/O. With different RocksDB instances storing raft logs and Region data, TiKV combines all the data write operations of raft logs and TiKV Regions into one I/O operation to improve the performance.

4.14.1.2 Region and Raft Consensus Algorithm

Data consistency between replicas of a Region is guaranteed by the Raft Consensus Algorithm. Only the leader of the Region can provide the writing service, and only when the data is written to the majority of replicas of a Region, the write operation succeeds.

When the size of a Region exceeds a threshold, which is 144 MB by default, TiKV splits it to two or more Regions. This operation guarantees the size of all the Regions in the cluster is nearly the same, which helps the PD component to balance Regions among nodes in a TiKV cluster. When the size of a Region is smaller than the threshold, TiKV merges the two smaller adjacent Regions into one Region.

When PD moves a replica from one TiKV node to another, it firstly adds a Learner replica on the target node, after the data in the Learner replica is nearly the same as that in the Leader replica, PD changes it to a Follower replica and removes the Follower replica on the source node.

Moving Leader replica from one node to another has a similar mechanism. The difference is that after the Learner replica becomes the Follower replica, there is a "Leader Transfer" operation in which the Follower replica actively proposes an election to elect itself as the Leader. Finally, the new Leader removes the old Leader replica in the source node.

4.14.2 Distributed Transaction

TiKV supports distributed transactions. Users (or TiDB) can write multiple key-value pairs without worrying about whether they belong to the same Region. TiKV uses two-phase commit to achieve ACID constraints. See TiDB Optimistic Transaction Model for details.

4.14.3 TiKV Coprocessor

TiDB pushes some data computation logic to TiKV Coprocessor. TiKV Coprocessor processes the computation for each Region. Each request sent to TiKV Coprocessor only involves the data of one Region.

4.15 TiDB Binlog

4.15.1 TiDB Binlog Cluster Overview

This document introduces the architecture and the deployment of the cluster version of TiDB Binlog.

TiDB Binlog is a tool used to collect binlog data from TiDB and provide near real-time backup and replication to downstream platforms.

TiDB Binlog has the following features:

- **Data replication:** replicate the data in the TiDB cluster to other databases
- **Real-time backup and restoration:** back up the data in the TiDB cluster and restore the TiDB cluster when the cluster fails

4.15.1.1 TiDB Binlog architecture

The TiDB Binlog architecture is as follows:

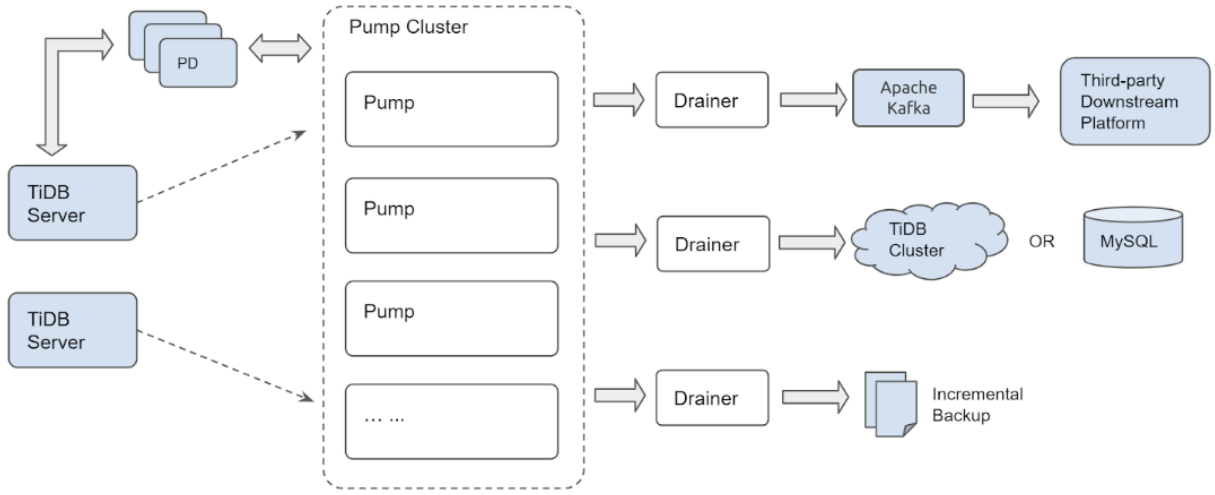

Figure 329: TiDB Binlog architecture

The TiDB Binlog cluster is composed of Pump and Drainer.

4.15.1.1.1 Pump

Pump is used to record the binlogs generated in TiDB, sort the binlogs based on the commit time of the transaction, and send binlogs to Drainer for consumption.

4.15.1.1.2 Drainer

Drainer collects and merges binlogs from each Pump, converts the binlog to SQL or data of a specific format, and replicates the data to a specific downstream platform.

4.15.1.1.3 `binlogctl` guide

`binlogctl` is an operations tool for TiDB Binlog with the following features:

- Obtaining the current `tso` of TiDB cluster
- Checking the Pump/Drainer state
- Modifying the Pump/Drainer state
- Pausing or closing Pump/Drainer

4.15.1.2 Main features

- Multiple Pumps form a cluster which can scale out horizontally
- TiDB uses the built-in Pump Client to send the binlog to each Pump
- Pump stores binlogs and sends the binlogs to Drainer in order

- Drainer reads binlogs of each Pump, merges and sorts the binlogs, and sends the binlogs downstream
- Drainer supports relay log. By the relay log, Drainer ensures that the downstream clusters are in a consistent state.

4.15.1.3 Notes

- You need to use TiDB v2.0.8-binlog, v2.1.0-rc.5 or a later version. Older versions of TiDB cluster are not compatible with the cluster version of TiDB Binlog.
- Drainer supports replicating binlogs to MySQL, TiDB, Kafka or local files. If you need to replicate binlogs to other Drainer unsuppored destinations, you can set Drainer to replicate the binlog to Kafka and read the data in Kafka for customized processing according to binlog consumer protocol. See Binlog Consumer Client User Guide.
- To use TiDB Binlog for recovering incremental data, set the config `db-type` to `file` ↪ (local files in the proto buffer format). Drainer converts the binlog to data in the specified proto buffer format and writes the data to local files. In this way, you can use Reparo to recover data incrementally.

  Pay attention to the value of `db-type`:

  - If your TiDB version is earlier than 2.1.9, set `db-type="pb"`.
  - If your TiDB version is 2.1.9 or later, set `db-type="file"` or `db-type="pb"`.

- If the downstream is MySQL, MariaDB, or another TiDB cluster, you can use sync-diff-inspector to verify the data after data replication.

4.15.2 TiDB Binlog Cluster Deployment

This document describes two methods of deploying TiDB Binlog:

- Deploy TiDB Binlog using TiDB Ansible
- Deploy TiDB Binlog using a Binary package

It is recommended to deploy TiDB Binlog using TiDB Ansible. If you just want to do a simple testing, you can deploy TiDB Binlog using a Binary package.

4.15.2.1 Hardware requirements

Pump and Drainer are deployed and operate on 64-bit universal hardware server platforms with Intel x86-64 architecture.

In environments of development, testing and production, the requirements on server hardware are as follows:

Service	The Number of Servers	CPU	Disk	Memory
Pump	3	8 core+	SSD, 200 GB+	16G
Drainer	1	8 core+	SAS, 100 GB+ (If binlogs are output as local files, the disk size depends on how long these files are retained.)	16G

4.15.2.2 Deploy TiDB Binlog using TiDB Ansible

4.15.2.2.1 Step 1: Download TiDB Ansible

1. Use the TiDB user account to log in to the central control machine and go to the `/home/tidb` directory. The information about the branch of TiDB Ansible and the corresponding TiDB version is as follows. If you have questions regarding which version to use, email to info@pingcap.com for more information or file an issue.

tidb-ansible branch	TiDB version	Note
release-3.0	3.0 stable	The latest 3.0 stable version. For use in production environments (recommended).

2. Use the following command to download the corresponding branch of TiDB Ansible

from the TiDB Ansible project on GitHub. The default folder name is `tidb-ansible`.

- Download the 3.0 branch:

```
git clone -b release-3.0 https://github.com/pingcap/tidb-ansible.
    ↪ git
```

4.15.2.2.2 Step 2: Deploy Pump

1. Modify the `tidb-ansible/inventory.ini` file.

    1. Set `enable_binlog = True` to start `binlog` of the TiDB cluster.

```
## binlog trigger
enable_binlog = True
```

    2. Add the deployment machine IPs for `pump_servers`.

```
## Binlog Part
[pump_servers]
172.16.10.72
172.16.10.73
172.16.10.74
```

    Pump retains the data of the latest 7 days by default. You can modify the value of the `gc` variable in the `tidb-ansible/conf/pump.yml` file (or `tidb-ansible/conf/pump-cluster.yml` in TiDB 3.0.2 or earlier versions) and remove the related comments:

```
global:
  # an integer value to control the expiry date of the binlog data,
      ↪  which indicates for how long (in days) the binlog data
      ↪ would be stored
  # must be bigger than 0
  # gc: 7
```

    Make sure the space of the deployment directory is sufficient for storing Binlog. For more details, see Configure the deployment directory. You can also set a separate deployment directory for Pump.

```
## Binlog Part
[pump_servers]
pump1 ansible_host=172.16.10.72 deploy_dir=/data1/pump
pump2 ansible_host=172.16.10.73 deploy_dir=/data2/pump
pump3 ansible_host=172.16.10.74 deploy_dir=/data3/pump
```

2. Deploy and start the TiDB cluster containing Pump.

   After configuring the `inventory.ini` file, you can choose one method from below to deploy the TiDB cluster.

   Method #1: Add Pump on the existing TiDB cluster.

   1. Deploy `pump_servers` and `node_exporters`.

```
ansible-playbook deploy.yml --tags=pump -l ${pump1_ip},${pump2_ip
    ↪ },[${alias1_name},${alias2_name}]
```

> **Note:**
> Do not add a space after the commas in the above command. Otherwise, an error is reported.

   2. Start `pump_servers`.

```
ansible-playbook start.yml --tags=pump
```

   3. Update and restart `tidb_servers`.

```
ansible-playbook rolling_update.yml --tags=tidb
```

   4. Update the monitoring data.

```
ansible-playbook rolling_update_monitor.yml --tags=prometheus
```

   Method #2: Deploy a TiDB cluster containing Pump from scratch.

   For how to use TiDB Ansible to deploy the TiDB cluster, see Deploy TiDB Using TiDB Ansible.

3. Check the Pump status.

   Use `binlogctl` to check the Pump status. Change the `pd-urls` parameter to the PD address of the cluster. If `State` is `online`, Pump is started successfully.

```
cd /home/tidb/tidb-ansible &&
resources/bin/binlogctl -pd-urls=http://172.16.10.72:2379 -cmd pumps
```

```
INFO[0000] pump: {NodeID: ip-172-16-10-72:8250, Addr:
    ↪ 172.16.10.72:8250, State: online, MaxCommitTS:
    ↪ 403051525690884099, UpdateTime: 2018-12-25 14:23:37 +0800 CST}
INFO[0000] pump: {NodeID: ip-172-16-10-73:8250, Addr:
    ↪ 172.16.10.73:8250, State: online, MaxCommitTS:
    ↪ 403051525703991299, UpdateTime: 2018-12-25 14:23:36 +0800 CST}
INFO[0000] pump: {NodeID: ip-172-16-10-74:8250, Addr:
    ↪ 172.16.10.74:8250, State: online, MaxCommitTS:
    ↪ 403051525717360643, UpdateTime: 2018-12-25 14:23:35 +0800 CST}
```

4.15.2.2.3 Step 3: Deploy Drainer

1. Obtain the value of `initial_commit_ts`.

   When Drainer starts for the first time, the timestamp information `initial_commit_ts` is required.

   - In TiDB 3.0.6 or later versions, if the replication is started from the latest time point, you just need to set `initial_commit_ts` to `-1`. Otherwise, you need to use `binlogctl` to get the the most recent timestamp information `initial_commit_ts` ↪ . Refer to the following method to obtain the latest timestamp.

   ```
   cd /home/tidb/tidb-ansible &&
   resources/bin/binlogctl -pd-urls=http://127.0.0.1:2379 -cmd
      ↪ generate_meta
   ```

   ```
   INFO[0000] [pd] create pd client with endpoints [http
      ↪ ://192.168.199.118:32379]
   INFO[0000] [pd] leader switches to: http://192.168.199.118:32379,
      ↪ previous:
   INFO[0000] [pd] init cluster id 6569368151110378289
   2018/06/21 11:24:47 meta.go:117: [info] meta: &{CommitTS
      ↪ :400962745252184065}
   ```

   This command outputs `meta: &{CommitTS:400962745252184065}`, and the value of `CommitTS` is the needed value of the `initial-commit-ts`.

   - If the downstream database is MySQL or TiDB, to ensure data integrity, you need to perform full data backup and recovery. In this case, the value of `initial_commit_ts` must be the timestamp information of the full backup.

     If you use mydumper to perform full data backup, you can get the timestamp by referring to the `Pos` field in the metadata file from the export directory. An example of the metadata file is as follows:

     ```
     Started dump at: 2019-12-30 13:25:41
     SHOW MASTER STATUS:
            Log: tidb-binlog
            Pos: 413580274257362947
            GTID:
     Finished dump at: 2019-12-30 13:25:41
     ```

2. Modify the `tidb-ansible/inventory.ini` file.

   Add the deployment machine IPs for `drainer_servers`. Set `initial_commit_ts` to the value you have obtained, which is only used for the initial start of Drainer.

   - Assume that the downstream is MySQL with the alias `drainer_mysql`:

```
[drainer_servers]
drainer_mysql ansible_host=172.16.10.71 initial_commit_ts
    ↪ ="402899541671542785"
```

- Assume that the downstream is `file` with the alias `drainer_file`:

```
[drainer_servers]
drainer_file ansible_host=172.16.10.71 initial_commit_ts
    ↪ ="402899541671542785"
```

3. Modify the configuration file.

- Assume that the downstream is MySQL:

```
cd /home/tidb/tidb-ansible/conf &&
cp drainer-cluster.toml drainer_mysql_drainer.toml &&
vi drainer_mysql_drainer.toml
```

> **Note:**
> Name the configuration file as `alias_drainer.toml`. Otherwise, the customized configuration file cannot be found during the deployment process. Note that in v3.0.0 and v3.0.1, you should name the configuration file as `alias_drainer-cluster.toml`.

Set `db-type` to `mysql` and configure the downstream MySQL information:

```
# downstream storage, equal to --dest-db-type
# Valid values are "mysql", "file", "tidb", and "kafka".
db-type = "mysql"

# the downstream MySQL protocol database
[syncer.to]
host = "172.16.10.72"
user = "root"
password = "123456"
port = 3306
# Time and size limits for flash batch write
```

- Assume that the downstream is incremental backup data:

```
cd /home/tidb/tidb-ansible/conf &&
cp drainer-cluster.toml drainer_file_drainer.toml &&
vi drainer_file_drainer.toml
```

Set `db-type` to `file`.

```
# downstream storage, equal to --dest-db-type
# Valid values are "mysql", "file", "tidb", and "kafka".
db-type = "file"

# Uncomment this if you want to use "file" as "db-type".
[syncer.to]
# default data directory: "{{ deploy_dir }}/data.drainer"
dir = "data.drainer"
```

4. Deploy Drainer.

```
ansible-playbook deploy_drainer.yml
```

5. Start Drainer.

```
ansible-playbook start_drainer.yml
```

4.15.2.3 Deploy TiDB Binlog using a Binary package

4.15.2.3.1 Download the official Binary package

Run the following commands to download the packages:

```
version="v3.0" for latest stable release of TiDB 3.0
wget https://download.pingcap.org/tidb-v3.0-linux-amd64.{tar.gz,sha256}
```

Check the file integrity. If the result is OK, the file is correct.

```
sha256sum -c tidb-v3.0-linux-amd64.sha256
```

For TiDB v2.1.0 GA or later versions, Pump and Drainer are already included in the TiDB download package. For other TiDB versions, you need to download Pump and Drainer separately using the following command:

```
version="latest" for nightly builds &&
wget https://download.pingcap.org/tidb-latest-linux-amd64.{tar.gz,sha256}
```

Check the file integrity. If the result is OK, the file is correct.

```
sha256sum -c tidb-latest-linux-amd64.sha256
```

4.15.2.3.2 The usage example

Assuming that you have three PD nodes, one TiDB node, two Pump nodes, and one Drainer node, the information of each node is as follows:

Node	IP
TiDB	192.168.0.10
PD1	192.168.0.16
PD2	192.168.0.15
PD3	192.168.0.14
Pump	192.168.0.11
Pump	192.168.0.12
Drainer	192.168.0.13

The following part shows how to use Pump and Drainer based on the nodes above.

1. Deploy Pump using the binary.

    - To view the command line parameters of Pump, execute `./bin/pump -help`:

```
Usage of Pump:
-L string
    the output information level of logs: debug, info, warn, error,
        ↪  fatal ("info" by default)
-V
    the print version information
-addr string
    the RPC address through which Pump provides the service (-addr=
        ↪ "192.168.0.11:8250")
-advertise-addr string
    the RPC address through which Pump provides the external
        ↪ service (-advertise-addr="192.168.0.11:8250")
-config string
    the path of the configuration file. If you specify the
        ↪ configuration file, Pump reads the configuration in the
        ↪ configuration file first. If the corresponding
        ↪ configuration also exits in the command line parameters,
        ↪ Pump uses the configuration of the command line
        ↪ parameters to cover that of the configuration file.
-data-dir string
    the path where the Pump data is stored
-gc int
    the number of days to retain the data in Pump ("7" by default)
-heartbeat-interval int
    the interval of the heartbeats Pump sends to PD (in seconds)
-log-file string
    the file path of logs
-log-rotate string
    the switch frequency of logs (hour/day)
```

```
-metrics-addr string
    the Prometheus Pushgateway address. If not set, it is forbidden
        ↪  to report the monitoring metrics.
-metrics-interval int
    the report frequency of the monitoring metrics ("15" by default
        ↪ , in seconds)
-node-id string
    the unique ID of a Pump node. If you do not specify this ID,
        ↪ the system automatically generates an ID based on the
        ↪ host name and listening port.
-pd-urls string
    the address of the PD cluster nodes (-pd-urls="http
        ↪ ://192.168.0.16:2379,http://192.168.0.15:2379,http
        ↪ ://192.168.0.14:2379")
-fake-binlog-interval int
    the frequency at which a Pump node generates fake binlog ("3"
        ↪ by default, in seconds)
```

- Taking deploying Pump on "192.168.0.11" as an example, the Pump configuration file is as follows:

```
# Pump Configuration

# the bound address of Pump
addr = "192.168.0.11:8250"

# the address through which Pump provides the service
advertise-addr = "192.168.0.11:8250"

# the number of days to retain the data in Pump ("7" by default)
gc = 7

# the directory where the Pump data is stored
data-dir = "data.pump"

# the interval of the heartbeats Pump sends to PD (in seconds)
heartbeat-interval = 2

# the address of the PD cluster nodes (each separated by a comma
    ↪ with no whitespace)
pd-urls = "http://192.168.0.16:2379,http://192.168.0.15:2379,http
    ↪ ://192.168.0.14:2379"

# [security]
# This section is generally commented out if no special security
```

```
   ↪ settings are required.
# The file path containing a list of trusted SSL CAs connected to
   ↪ the cluster.
# ssl-ca = "/path/to/ca.pem"
# The path to the X509 certificate in PEM format that is connected
   ↪ to the cluster.
# ssl-cert = "/path/to/drainer.pem"
# The path to the X509 key in PEM format that is connected to the
   ↪ cluster.
# ssl-key = "/path/to/drainer-key.pem"

# [storage]
# Set to true (by default) to guarantee reliability by ensuring
   ↪ binlog data is flushed to the disk.
# sync-log = true

# When the available disk capacity is less than the set value, Pump
   ↪  stops writing data.
# 42 MB -> 42000000, 42 mib -> 44040192
# default: 10 gib
# stop-write-at-available-space = "10 gib"
# The LSM DB settings embedded in Pump. Unless you know this part
   ↪ well, it is usually commented out.
# [storage.kv]
# block-cache-capacity = 8388608
# block-restart-interval = 16
# block-size = 4096
# compaction-L0-trigger = 8
# compaction-table-size = 67108864
# compaction-total-size = 536870912
# compaction-total-size-multiplier = 8.0
# write-buffer = 67108864
# write-L0-pause-trigger = 24
# write-L0-slowdown-trigger = 17
```

- The example of starting Pump:

```
./bin/pump -config pump.toml
```

  If the command line parameters is the same with the configuration file parameters, the values of command line parameters are used.

2. Deploy Drainer using binary.

   - To view the command line parameters of Drainer, execute `./bin/drainer -help`:

```
Usage of Drainer:
-L string
    the output information level of logs: debug, info, warn, error,
        ↪  fatal ("info" by default)
-V
    the print version information
-addr string
    the address through which Drainer provides the service (-addr="
        ↪ 192.168.0.13:8249")
-c int
    the number of the concurrency of the downstream for replication
        ↪ . The bigger the value, the better throughput performance
        ↪  of the concurrency ("1" by default).
-cache-binlog-count int
    the limit on the number of binlog items in the cache ("8" by
        ↪ default)
    If a large single binlog item in the upstream causes OOM in
        ↪ Drainer, try to lower the value of this parameter to
        ↪ reduce memory usage.
-config string
    the directory of the configuration file. Drainer reads the
        ↪ configuration file first.
    If the corresponding configuration exists in the command line
        ↪ parameters, Drainer uses the configuration of the command
        ↪  line parameters to cover that of the configuration file.
-data-dir string
    the directory where the Drainer data is stored ("data.drainer"
        ↪ by default)
-dest-db-type string
    the downstream service type of Drainer
    The value can be "mysql", "tidb", "kafka", and "file". ("mysql"
        ↪  by default)
-detect-interval int
    the interval of checking the online Pump in PD ("10" by default
        ↪ , in seconds)
-disable-detect
    whether to disable the conflict monitoring
-disable-dispatch
    whether to disable the SQL feature of splitting a single binlog
        ↪  file. If it is set to "true", each binlog file is
        ↪ restored to a single transaction for replication based on
        ↪  the order of binlogs.
    It is set to "False", when the downstream is MySQL.
-ignore-schemas string
```

```
    the db filter list ("INFORMATION_SCHEMA,PERFORMANCE_SCHEMA,
        ↪ mysql,test" by default)
    It does not support the Rename DDL operation on tables of `
        ↪ ignore schemas`.
-initial-commit-ts
    If Drainer does not have the related breakpoint information,
        ↪ you can configure the related breakpoint information
        ↪ using this parameter. (In TiDB 3.0.6 or later versions,
        ↪ the value of this parameter is `-1` by default; Before
        ↪ TiDB 3.0.6, the value is `0` by default.)
    If the value of this parameter is `-1`, Drainer automatically
        ↪ obtains the latest timestamp from PD.
-log-file string
    the path of the log file
-log-rotate string
    the switch frequency of log files, hour/day
-metrics-addr string
    the Prometheus Pushgateway address
    It it is not set, the monitoring metrics are not reported.
-metrics-interval int
    the report frequency of the monitoring metrics ("15" by default
        ↪ , in seconds)
-node-id string
    the unique ID of a Drainer node. If you do not specify this ID,
        ↪  the system automatically generates an ID based on the
        ↪ host name and listening port.
-pd-urls string
    the address of the PD cluster nodes (-pd-urls="http
        ↪ ://192.168.0.16:2379,http://192.168.0.15:2379,http
        ↪ ://192.168.0.14:2379")
-safe-mode
    Whether to enable safe mode so that data can be written into
        ↪ the downstream MySQL/TiDB repeatedly.
    This mode replaces the `INSERT` statement with the `REPLACE`
        ↪ statement and splits the `UPDATE` statement into `DELETE`
        ↪  plus `REPLACE`.
-txn-batch int
    the number of SQL statements of a transaction which are output
        ↪ to the downstream database ("1" by default)
```

- Taking deploying Drainer on "192.168.0.13" as an example, the Drainer configuration file is as follows:

```
# Drainer Configuration.
```

```
# the address through which Drainer provides the service
    ↪ ("192.168.0.13:8249")
addr = "192.168.0.13:8249"

# the address through which Drainer provides the external service
advertise-addr = "192.168.0.13:8249"

# the interval of checking the online Pump in PD ("10" by default,
    ↪ in seconds)
detect-interval = 10

# the directory where the Drainer data is stored "data.drainer" by
    ↪ default)
data-dir = "data.drainer"

# the address of the PD cluster nodes (each separated by a comma
    ↪ with no whitespace)
pd-urls = "http://192.168.0.16:2379,http://192.168.0.15:2379,http
    ↪ ://192.168.0.14:2379"

# the directory of the log file
log-file = "drainer.log"

# Drainer compresses the data when it gets the binlog from Pump.
    ↪ The value can be "gzip". If it is not configured, it will
    ↪ not be compressed
# compressor = "gzip"

# [security]
# This section is generally commented out if no special security
    ↪ settings are required.
# The file path containing a list of trusted SSL CAs connected to
    ↪ the cluster.
# ssl-ca = "/path/to/ca.pem"
# The path to the X509 certificate in PEM format that is connected
    ↪ to the cluster.
# ssl-cert = "/path/to/pump.pem"
# The path to the X509 key in PEM format that is connected to the
    ↪ cluster.
# ssl-key = "/path/to/pump-key.pem"

# Syncer Configuration
[syncer]
# If the item is set, the sql-mode will be used to parse the DDL
    ↪ statement.
```

```
# If the downstream database is MySQL or TiDB, then the downstream
    ↪ sql-mode
# is also set to this value.
# sql-mode = "STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION"

# the number of SQL statements of a transaction that are output to
    ↪ the downstream database ("20" by default)
txn-batch = 20

# the number of the concurrency of the downstream for replication.
    ↪ The bigger the value,
# the better throughput performance of the concurrency ("16" by
    ↪ default)
worker-count = 16

# whether to disable the SQL feature of splitting a single binlog
    ↪ file. If it is set to "true",
# each binlog file is restored to a single transaction for
    ↪ replication based on the order of binlogs.
# If the downstream service is MySQL, set it to "False".
disable-dispatch = false

# In safe mode, data can be written into the downstream MySQL/TiDB
    ↪ repeatedly.
# This mode replaces the `INSERT` statement with the `REPLACE`
    ↪ statement and replaces the `UPDATE` statement with `DELETE`
    ↪ plus `REPLACE` statements.
safe-mode = false

# the downstream service type of Drainer ("mysql" by default)
# Valid value: "mysql", "file", "tidb", and "kafka".
db-type = "mysql"

# If `commit ts` of the transaction is in the list, the transaction
    ↪  is filtered and not replicated to the downstream.
ignore-txn-commit-ts = []

# the db filter list ("INFORMATION_SCHEMA,PERFORMANCE_SCHEMA,mysql,
    ↪ test" by default)
# Does not support the Rename DDL operation on tables of `ignore
    ↪ schemas`.
ignore-schemas = "INFORMATION_SCHEMA,PERFORMANCE_SCHEMA,mysql"

# `replicate-do-db` has priority over `replicate-do-table`. When
    ↪ they have the same `db` name,
```

```
# regular expressions are supported for configuration.
# The regular expression should start with "~".

# replicate-do-db = ["~^b.*","s1"]

# [syncer.relay]
# It saves the directory of the relay log. The relay log is not
   ↪ enabled if the value is empty.
# The configuration only comes to effect if the downstream is TiDB
   ↪ or MySQL.
# log-dir = ""
# the maximum size of each file
# max-file-size = 10485760

# [[syncer.replicate-do-table]]
# db-name ="test"
# tbl-name = "log"

# [[syncer.replicate-do-table]]
# db-name ="test"
# tbl-name = "~^a.*"

# Ignore the replication of some tables
# [[syncer.ignore-table]]
# db-name = "test"
# tbl-name = "log"

# the server parameters of the downstream database when `db-type`
   ↪ is set to "mysql"
[syncer.to]
host = "192.168.0.13"
user = "root"
password = ""
# `encrypted_password` is encrypted using `./binlogctl -cmd encrypt
   ↪  -text string`.
# When `encrypted_password` is not empty, the `password` above will
   ↪  be ignored.
encrypted_password = ""
port = 3306

[syncer.to.checkpoint]
# When the checkpoint type is "mysql" or "tidb", this option can be
   ↪  enabled to change the database that saves the checkpoint
# schema = "tidb_binlog"
# Currently only the "mysql" and "tidb" checkpoint types are
```

```
    ↪ supported
# You can remove the comment tag to control where to save the
    ↪ checkpoint
# The default method of saving the checkpoint for the downstream db
    ↪ -type:
# mysql/tidb -> in the downstream MySQL or TiDB database
# file/kafka -> file in `data-dir`
# type = "mysql"
# host = "127.0.0.1"
# user = "root"
# password = ""
# `encrypted_password` is encrypted using `./binlogctl -cmd encrypt
    ↪  -text string`.
# When `encrypted_password` is not empty, the `password` above will
    ↪  be ignored.
# encrypted_password = ""
# port = 3306

# the directory where the binlog file is stored when `db-type` is
    ↪ set to `file`
# [syncer.to]
# dir = "data.drainer"

# the Kafka configuration when `db-type` is set to "kafka"
# [syncer.to]
# only one of kafka-addrs and zookeeper-addrs is needed. If both
    ↪ are present, the program gives priority to the kafka address
    ↪  in zookeeper.
# zookeeper-addrs = "127.0.0.1:2181"
# kafka-addrs = "127.0.0.1:9092"
# kafka-version = "0.8.2.0"
# kafka-max-messages = 1024

# the topic name of the Kafka cluster that saves the binlog data.
    ↪ The default value is <cluster-id>_obinlog
# To run multiple Drainers to replicate data to the same Kafka
    ↪ cluster, you need to set different `topic-name`s for each
    ↪ Drainer.
# topic-name = ""
```

- Starting Drainer:

> **Note:**
> If the downstream is MySQL/TiDB, to guarantee the data integrity,

you need to obtain the `initial-commit-ts` value and make a full backup of the data and restore the data before the initial start of Drainer. For details, see Deploy Drainer.

When Drainer is started for the first time, use the `initial-commit-ts` parameter.

```
./bin/drainer -config drainer.toml -initial-commit-ts {initial-
    ↪ commit-ts}
```

If the command line parameter and the configuration file parameter are the same, the parameter value in the command line is used.

3. Starting TiDB server:

   - After starting Pump and Drainer, start TiDB server with binlog enabled by adding this section to your config file for TiDB server:

     ```
     [binlog]
     enable=true
     ```

   - TiDB server will obtain the addresses of registered Pumps from PD and will stream data to all of them. If there are no registered Pump instances, TiDB server will refuse to start or will block starting until a Pump instance comes online.

Note:

- When TiDB is running, you need to guarantee that at least one Pump is running normally.
- To enable the TiDB Binlog service in TiDB server, use the `-enable-binlog` startup parameter in TiDB, or add enable=true to the binlog section of the TiDB server configuration file.
- Make sure that the TiDB Binlog service is enabled in all TiDB instances in a same cluster, otherwise upstream and downstream data inconsistency might occur during data replication. If you want to temporarily run a TiDB instance where the TiDB Binlog service is not enabled, set `run_ddl=false` in the TiDB configuration file.
- Drainer does not support the `rename` DDL operation on the table of `ignore schemas` (the schemas in the filter list).
- If you want to start Drainer in an existing TiDB cluster, generally you need to make a full backup of the cluster data, obtain **snapshot timestamp**, import the data to the target database, and then start Drainer to replicate the incremental data from corresponding **snapshot timestamp**.

- When the downstream database is TiDB or MySQL, ensure that the `sql_mode` in the upstream and downstream databases are consistent. In other words, the `sql_mode` should be the same when each SQL statement is executed in the upstream and replicated to the downstream. You can execute the `select @@sql_mode;` statement in the upstream and downstream respectively to compare `sql_mode`.
- When a DDL statement is supported in the upstream but incompatible with the downstream, Drainer fails to replicate data. An example is to replicate the `CREATE TABLE t1(a INT)ROW_FORMAT=FIXED;` statement when the downstream database MySQL uses the InnoDB engine. In this case, you can configure skipping transactions in Drainer, and manually execute compatible statements in the downstream database.

4.15.3 TiDB Binlog Cluster Operations

This document introduces the following TiDB Binlog cluster operations:

- The state of a Pump and Drainer nodes
- Starting or exiting a Pump or Drainer process
- Managing the TiDB Binlog cluster by using the binlogctl tool or by directly performing SQL operations in TiDB

4.15.3.1 Pump or Drainer state

Pump or Drainer state description:

- `online`: running normally
- `pausing`: in the pausing process
- `paused`: has been stopped
- `closing`: in the offline process
- `offline`: has been offline

Note:

The state information of a Pump or Drainer node is maintained by the service itself and is regularly updated to the Placement Driver (PD).

4.15.3.2 Starting and exiting a Pump or Drainer process

4.15.3.2.1 Pump

- Starting: When started, the Pump node notifies all Drainer nodes in the `online` state. If the notification is successful, the Pump node sets its state to `online`. Otherwise, the Pump node reports an error, sets its state to `paused` and exits the process.
- Exiting: The Pump node enters the `paused` or `offline` state before the process is exited normally; if the process is exited abnormally (caused by the `kill -9` command, process panic, crash), the node is still in the `online` state.
  - Pause: You can pause a Pump process by using the `kill` command (not `kill -9` ↪), pressing Ctrl+C or using the `pause-pump` command in the binlogctl tool. After receiving the pause instruction, the Pump node sets its state to `pausing`, stops receiving binlog write requests and stops providing binlog data to Drainer nodes. After all threads are safely exited, the Pump node updates its state to `paused` and exits the process.
  - Offline: You can close a Pump process only by using the `offline-pump` command in the binlogctl tool. After receiving the offline instruction, the Pump node sets its state to `closing` and stops receiving the binlog write requests. The Pump node continues providing binlog to Drainer nodes until all binlog data is consumed by Drainer nodes. Then, the Pump node sets its state to `offline` and exits the process.

4.15.3.2.2 Drainer

- Starting: When started, the Drainer node sets its state to `online` and tries to pull binlogs from all Pump nodes which are not in the `offline` state. If it fails to get the binlogs, it keeps trying.
- Exiting: The Drainer node enters the `paused` or `offline` state before the process is exited normally; if the process is exited abnormally (caused by `kill -9`, process panic, crash), the Drainer node is still in the `online` state.
  - Pause: You can pause a Drainer process by using the `kill` command (not `kill` ↪ `-9`), pressing Ctrl+C or using the `pause-drainer` command in the binlogctl tool. After receiving the pause instruction, the Drainer node sets its state to `pausing` and stops pulling binlogs from Pump nodes. After all threads are safely exited, the Drainer node sets its state to `paused` and exits the process.
  - Offline: You can close a Drainer process only by using the `offline-drainer` command in the binlogctl tool. After receiving the offline instruction, the Drainer node sets its state to `closing` and stops pulling binlogs from Pump nodes. After all threads are safely exited, the Drainer node updates its state to `offline` and exits the process.

For how to pause, close, check, and modify the state of Drainer, see the binlogctl guide as follows.

4.15.3.3 `binlogctl` guide

`binlogctl` is an operations tool for TiDB Binlog with the following features:

- Checking the state of Pump or Drainer
- Pausing or closing Pump or Drainer
- Handling the abnormal state of Pump or Drainer

4.15.3.3.1 Usage scenarios of `binlogctl`

- An error occurs during data replication and you need to check the running status and state of Pump or Drainer.
- While maintaining the cluster, you need to pause or close Pump or Drainer.
- Pump or Drainer process is exited abnormally, but the node state is not updated or is unexpected, which influences the application.

4.15.3.3.2 Download `binlogctl`

Your distribution of TiDB or TiDB Binlog might already include binlogctl. If not, download `binlogctl`:

```
wget https://download.pingcap.org/tidb-{version}-linux-amd64.tar.gz && \
wget https://download.pingcap.org/tidb-{version}-linux-amd64.sha256
```

The following command checks the file integrity. If the result is OK, the file is correct.

```
sha256sum -c tidb-{version}-linux-amd64.sha256
```

For TiDB v2.1.0 GA or later versions, binlogctl is already included in the TiDB download package. For earlier versions, you need to download binlogctl separately.

```
wget https://download.pingcap.org/tidb-enterprise-tools-latest-linux-amd64.
    ↪ tar.gz && \
wget https://download.pingcap.org/tidb-enterprise-tools-latest-linux-amd64.
    ↪ sha256
```

The following command checks the file integrity. If the result is OK, the file is correct.

```
sha256sum -c tidb-enterprise-tools-latest-linux-amd64.sha256
```

4.15.3.3.3 `binlogctl` usage description

Command line parameters:

```
Usage of binlogctl:
-V
    Outputs the binlogctl version information
-cmd string
    the command mode, including "generate_meta" (deprecated), "pumps", "
        ↪ drainers", "update-pump" ,"update-drainer", "pause-pump", "pause-
        ↪ drainer", "offline-pump", and "offline-drainer"
-data-dir string
    the file path where the checkpoint file of Drainer is stored ("
        ↪ binlog_position" by default) (deprecated)
-node-id string
    ID of Pump or Drainer
-pd-urls string
    the address of PD. If multiple addresses exist, use "," to separate each
        ↪  ("http://127.0.0.1:2379" by default)
-ssl-ca string
    the file path of SSL CAs
-ssl-cert string
    the file path of the X509 certificate file in the PEM format
-ssl-key string
    the file path of X509 key file of the PEM format
-time-zone string
    If a time zone is set, the corresponding time of the obtained `tso` is
        ↪ printed in the "generate_meta" mode. For example, "Asia/Shanghai"
        ↪ is the CST time zone and "Local" is the local time zone
-show-offline-nodes
    used with the `-cmd pumps` or `-cmd drainers` command. The two commands
        ↪ do not show the offline node by default unless this parameter is
        ↪ explicitly specified
```

Command example:

- Check the state of all the Pump or Drainer nodes:

  Set `cmd` as `pumps` or `drainers` to check the state of all the Pump or Drainer nodes. For example,

```
bin/binlogctl -pd-urls=http://127.0.0.1:2379 -cmd pumps
```

```
[2019/04/28 09:29:59.016 +00:00] [INFO] [nodes.go:48] ["query node"] [
    ↪ type=pump] [node="{NodeID: 1.1.1.1:8250, Addr: pump:8250, State:
    ↪ online, MaxCommitTS: 408012403141509121, UpdateTime: 2019-04-28
    ↪ 09:29:57 +0000 UTC}"]
```

```
bin/binlogctl -pd-urls=http://127.0.0.1:2379 -cmd drainers
```

```
[2019/04/28 09:29:59.016 +00:00] [INFO] [nodes.go:48] ["query node"] [
    ↪ type=drainer] [node="{NodeID: 1.1.1.1:8249, Addr: 1.1.1.1:8249,
    ↪ State: online, MaxCommitTS: 408012403141509121, UpdateTime:
    ↪ 2019-04-28 09:29:57 +0000 UTC}"]
```

- Pause or close Pump or Drainer:

  binlogctl provides the following commands to pause or close services:

cmd	Description	Example
pause-pump	Pause Pump	`bin/binlogctl -pd-urls=http://127.0.0.1:2379 -cmd pause-pump -node-id ip-127-0-0-1:8250`
pause-drainer	Pause Drainer	`bin/binlogctl -pd-urls=http://127.0.0.1:2379 -cmd pause-drainer -node-id ip-127-0-0-1:8249`
offline-pump	Close Pump	`bin/binlogctl -pd-urls=http://127.0.0.1:2379 -cmd offline-pump -node-id ip-127-0-0-1:8250`
offline-drainer	Close Drainer	`bin/binlogctl -pd-urls=http://127.0.0.1:2379 -cmd offline-drainer -node-id ip-127-0-0-1:8249`

  binlogctl sends the HTTP request to the Pump or Drainer node. After receiving the request, the node executes the corresponding exiting procedures.

- Modify the state of a Pump or Drainer node in abnormal situations

  When a Pump or Drainer node runs normally or when it is paused or closed in the normal process, it is in the right state. But in some abnormal situations, the Pump or Drainer node cannot correctly maintain its state, which can influence data replication tasks. In these situations, use the binlogctl tool to repair the state information.

  Set `cmd` to `update-pump` or `update-drainer` to update the state of a Pump or Drainer node. The state can be `paused` or `offline`. For example:

```
bin/binlogctl -pd-urls=http://127.0.0.1:2379 -cmd update-pump -node-id
    ↪ ip-127-0-0-1:8250 -state paused
```

> **Note:**
> When a Pump or Drainer node runs normally, it regularly updates its state to PD. But the above command is used to directly modify the Pump or Drainer state saved in PD, so do not use the command when the Pump or Drainer node runs normally. Refer to TiDB Binlog FAQ to see in what situation you need to use it.

4.15.3.4 Use SQL statements to manage Pump or Drainer

To view or modify binlog related states, execute corresponding SQL statements in TiDB.

- Check whether binlog is enabled:

```
show variables like "log_bin";
```

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       |  ON   |
+---------------+-------+
```

When the Value is `ON`, it means that the binlog is enabled.

- Check the status of all the Pump or Drainer nodes:

```
show pump status;
```

```
+--------|----------------|--------|--------------------|--------------------|
   ↪
| NodeID |   Address    | State |   Max_Commit_Ts  |    Update_Time     |
+--------|----------------|--------|--------------------|--------------------|
   ↪
| pump1 | 127.0.0.1:8250 | Online | 408553768673342237 | 2019-05-01
   ↪ 00:00:01 |
+--------|----------------|--------|--------------------|--------------------|
   ↪
| pump2 | 127.0.0.2:8250 | Online | 408553768673342335 | 2019-05-01
   ↪ 00:00:02 |
+--------|----------------|--------|--------------------|--------------------|
   ↪
```

```
show drainer status;
```

```
+----------|----------------|--------|--------------------|--------------------|
   ↪
|  NodeID |     Address    | State |   Max_Commit_Ts  |    Update_Time     |
+----------|----------------|--------|--------------------|--------------------|
   ↪
| drainer1 | 127.0.0.3:8249 | Online | 408553768673342532 | 2019-05-01
   ↪ 00:00:03 |
+----------|----------------|--------|--------------------|--------------------|
   ↪
| drainer2 | 127.0.0.4:8249 | Online | 408553768673345531 | 2019-05-01
   ↪ 00:00:04 |
```

```
+----------|---------------|--------|--------------------|-----------------------|
    ↪
```

- Modify the states of a Pump or Drainer node in abnormal situations

```
change pump to node_state ='paused' for node_id 'pump1';
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
change drainer to node_state ='paused' for node_id 'drainer1';
```

```
Query OK, 0 rows affected (0.01 sec)
```

Executing the above SQL statements works the same as the `update-pump` or `update ↪ -drainer` commands in binlogctl. Use the above SQL statements **only** when the Pump or Drainer node is in abnormal situations.

Note:

- Checking whether binlog is enabled and the running status of Pump or Drainer is supported in TiDB v2.1.7 and later versions.
- Modifying the status of Pump or Drainer is supported in TiDB v3.0.0-rc.1 and later versions. This feature only supports modifying the status of Pump or Drainer nodes stored in the Placement Driver (PD). To pause or close the node, use the `binlogctl` tool.

4.15.4 TiDB Binlog Monitoring

After you have deployed TiDB Binlog using TiDB Ansible successfully, you can go to the Grafana Web (default address: http://grafana_ip:3000, default account: admin, password: admin) to check the state of Pump and Drainer.

4.15.4.1 Monitoring metrics

TiDB Binlog consists of two components: Pump and Drainer. This section shows the monitoring metrics of Pump and Drainer.

4.15.4.1.1 Pump monitoring metrics

To understand the Pump monitoring metrics, check the following table:

Pump monitoring metrics	Description
Storage Size	Records the total disk space (capacity) and the available disk space (available)

Pump monitoring metrics	Description
Metadata	Records the biggest TSO (`gc_tso` ↪) of the binlog that each Pump node can delete, and the biggest commit TSO (`max_commit_tso` ↪) of the saved binlog

Pump monitoring metrics	Description
Write Binlog QPS by Instance	Shows QPS of writing binlog requests received by each Pump node
Write Binlog Latency	Records the latency time of each Pump node writing binlog

Pump monitoring metrics	Description
Storage Write Binlog Size	Shows the size of the binlog data written by Pump
Storage Write Binlog Latency	Records the latency time of the Pump storage module writing binlog

Pump monitoring metrics	Description
Pump Storage Error By Type	Records the number of errors encountered by Pump, counted based on the type of error
Query TiKV	The number of times that Pump queries the transaction status through TiKV

4.15.4.1.2 Drainer monitoring metrics

To understand the Drainer monitoring metrics, check the following table:

Drainer monitoring metrics	Description
Drainer monitoring metrics	Description
Checkpoint TSO	Shows the biggest TSO time of the binlog that Drainer has already replicated into the downstream. You can get the lag by using the current time to subtract the bin-

Drainer monitoring metrics	Description
Pump Handle TSO	Records the biggest TSO time among the binlog files that Drainer obtains from each Pump node
Pull Binlog QPS by Pump NodeID	Shows the QPS when Drainer obtains binlog from each Pump node

Drainer monitoring metrics	Description
95% Binlog Reach Duration By Pump	Records the delay from the time when binlog is written into Pump to the time when the binlog is obtained by Drainer

Drainer monitoring metrics	Description
Error By Type	Shows the number of errors encountered by Drainer, counted based on the type of error
SQL Query Time	Records the time it takes Drainer to execute the SQL statement in the downstream

Drainer monitoring metrics	Description
Drainer Event	Shows the number of various types of events, including “ddl”, “insert”, “delete”, “update”, “flush”, and “savepoint”

Drainer monitoring metrics	Description
Execute Time	Records the time it takes to write binlog into the downstream syncing module
95% Binlog Size	Shows the size of the binlog data that Drainer obtains from each Pump node

Drainer monitoring metrics	Description
DDL Job Count	Records the number of DDL statements handled by Drainer
Queue Size	Records the work queue size in Drainer

4.15.4.2 Alert rules

This section gives the alert rules for TiDB Binlog. According to the severity level, TiDB Binlog alert rules are divided into three categories (from high to low): emergency-level, critical-level and warning-level.

4.15.4.2.1 Emergency-level alerts

Emergency-level alerts are often caused by a service or node failure. Manual intervention is required immediately.

`binlog_pump_storage_error_count`

- Alert rule:

  `changes(binlog_pump_storage_error_count[1m])> 0`

- Description:

  Pump fails to write the binlog data to the local storage.

- Solution:

  Check whether an error exists in the `pump_storage_error` monitoring and check the Pump log to find the causes.

4.15.4.2.2 Critical-level alerts

For the critical-level alerts, a close watch on the abnormal metrics is required.

`binlog_drainer_checkpoint_high_delay`

- Alert rule:

  `(time()- binlog_drainer_checkpoint_tso / 1000)> 3600`

- Description:

  The delay of Drainer replication exceeds one hour.

- Solution:

  - Check whether it is too slow to obtain the data from Pump:

    You can check `handle tso` of Pump to get the time for the latest message of each Pump. Check whether a high latency exists for Pump and make sure the corresponding Pump is running normally.
  - Check whether it is too slow to replicate data in the downstream based on Drainer `event` and Drainer `execute latency`:
    * If Drainer `execute time` is too large, check the network bandwidth and latency between the machine with Drainer deployed and the machine with the target database deployed, and the state of the target database.
    * If Drainer `execute time` is not too large and Drainer `event` is too small, add `work count` and `batch` and retry.
  - If the two solutions above cannot work, contact support@pingcap.com.

4.15.4.2.3 Warning-level alerts

Warning-level alerts are a reminder for an issue or error.

`binlog_pump_write_binlog_rpc_duration_seconds_bucket`

- Alert rule:

  ```
  histogram_quantile(0.9, rate(binlog_pump_rpc_duration_seconds_bucket{
  ↪ method="WriteBinlog"}[5m]))> 1
  ```

- Description:

  It takes too much time for Pump to handle the TiDB request of writing binlog.

- Solution:

  - Verify the disk performance pressure and check the disk performance monitoring via `node exported`.
  - If both `disk latency` and `util` are low, contact support@pingcap.com.

`binlog_pump_storage_write_binlog_duration_time_bucket`

- Alert rule:

  ```
  histogram_quantile(0.9, rate(binlog_pump_storage_write_binlog_duration_time_bucket
  ↪ {type="batch"}[5m]))> 1
  ```

- Description:

  The time it takes for Pump to write the local binlog to the local disk.

- Solution:

  Check the state of the local disk of Pump and fix the problem.

`binlog_pump_storage_available_size_less_than_20G`

- Alert rule:

  ```
  binlog_pump_storage_storage_size_bytes{type="available"} < 20 * 1024 *
  ↪ 1024 * 1024
  ```

- Description:

  The available disk space of Pump is less than 20 GB.

- Solution:

  Check whether Pump `gc_tso` is normal. If not, adjust the GC time configuration of Pump or get the corresponding Pump offline.

`binlog_drainer_checkpoint_tso_no_change_for_1m`

- Alert rule:

  ```
  changes(binlog_drainer_checkpoint_tso[1m])< 1
  ```

- Description:

  Drainer `checkpoint` has not been updated for one minute.

- Solution:

  Check whether all the Pumps that are not offline are running normally.

`binlog_drainer_execute_duration_time_more_than_10s`

- Alert rule:

  ```
  histogram_quantile(0.9, rate(binlog_drainer_execute_duration_time_bucket
  ↪ [1m]))> 10
  ```

- Description:

  The transaction time it takes Drainer to replicate data to TiDB. If it is too large, the Drainer replication of data is affected.

- Solution:

  - Check the TiDB cluster state.
  - Check the Drainer log or monitor. If a DDL operation causes this problem, you can ignore it.

4.15.5 TiDB Binlog Configuration File

This document introduces the configuration items of TiDB Binlog.

4.15.5.1 Pump

This section introduces the configuration items of Pump. For the example of a complete Pump configuration file, see Pump Configuration.

4.15.5.1.1 addr

- Specifies the listening address of HTTP API in the format of `host:port`.
- Default value: `127.0.0.1:8250`

4.15.5.1.2 advertise-addr

- Specifies the externally accessible HTTP API address. This address is registered in PD in the format of `host:port`.
- Default value: `127.0.0.1:8250`

4.15.5.1.3 socket

- The Unix socket address that HTTP API listens to.
- Default value: “”

4.15.5.1.4 pd-urls

- Specifies the comma-separated list of PD URLs. If multiple addresses are specified, when the PD client fails to connect to one address, it automatically tries to connect to another address.
- Default value: `http://127.0.0.1:2379`

4.15.5.1.5 data-dir

- Specifies the directory where binlogs and their indexes are stored locally.
- Default value: `data.pump`

4.15.5.1.6 heartbeat-interval

- Specifies the heartbeat interval (in seconds) at which the latest status is reported to PD.
- Default value: `2`

4.15.5.1.7 gen-binlog-interval

- Specifies the interval (in seconds) at which data is written into fake binlog.
- Default value: `3`

4.15.5.1.8 gc

- Specifies the number of days (integer) that binlogs can be stored locally. Binlogs stored longer than the specified number of days are automatically deleted.
- Default value: `7`

4.15.5.1.9 log-file

- Specifies the path where log files are stored. If the parameter is set to an empty value, log files are not stored.
- Default value: “”

4.15.5.1.10 log-level

- Specifies the log level.
- Default value: `info`

4.15.5.1.11 node-id

- Specifies the Pump node ID. With this ID, this Pump process can be identified in the cluster.
- Default value: `hostname:port number`. For example, `node-1:8250`.

4.15.5.1.12 security

This section introduces configuration items related to security.

ssl-ca

- Specifies the file path of the trusted SSL certificate list or CA list. For example, `/path/to/ca.pem`.
- Default value: ""

ssl-cert

- Specifies the path of the X509 certificate file encoded in the Privacy Enhanced Mail (PEM) format. For example, `/path/to/pump.pem`.
- Default value: ""

ssl-key

- Specifies the path of the X509 key file encoded in the PEM format. For example, `/path/to/pump-key.pem`.
- Default value: ""

4.15.5.1.13 storage

This section introduces configuration items related to storage.

sync-log

- Specifies whether to use `fsync` after each **batch** write to binlog to ensure data safety.
- Default value: `true`

kv_chan_cap

- Specifies the number of write requests that the buffer can store before Pump receives these requests.
- Default value: `1048576` (that is, 2 to the power of 20)

slow_write_threshold

- The threshold (in seconds). If it takes longer to write a single binlog file than this specified threshold, the write is considered slow write and `"take a long time to write binlog"` is output in the log.
- Default value: `1`

stop-write-at-available-space

- Binlog write requests is no longer accepted when the available storage space is below this specified value. You can use the format such as `900 MB`, `5 GB`, and `12 GiB` to specify the storage space. If there is more than one Pump node in the cluster, when a Pump node refuses a write request because of the insufficient space, TiDB will automatically write binlogs to other Pump nodes.
- Default value: `10 GiB`

kv

Currently the storage of Pump is implemented based on GoLevelDB. Under `storage` there is also a `kv` subgroup that is used to adjust the GoLevel configuration. The supported configuration items are shown as below:

- block-cache-capacity
- block-restart-interval
- block-size
- compaction-L0-trigger
- compaction-table-size
- compaction-total-size
- compaction-total-size-multiplier
- write-buffer
- write-L0-pause-trigger
- write-L0-slowdown-trigger

For the detailed description of the above items, see GoLevelDB Document.

4.15.5.2 Drainer

This section introduces the configuration items of Drainer. For the example of a complete Drainer configuration file, see Drainer Configuration

4.15.5.2.1 addr

- Specifies the listening address of HTTP API in the format of `host:port`.
- Default value: `127.0.0.1:8249`

4.15.5.2.2 advertise-addr

- Specifies the externally accessible HTTP API address. This address is registered in PD in the format of `host:port`.
- Default value: `127.0.0.1:8249`

4.15.5.2.3 log-file

- Specifies the path where log files are stored. If the parameter is set to an empty value, log files are not stored.
- Default value: “”

4.15.5.2.4 log-level

- Specifies the log level.
- Default value: `info`

4.15.5.2.5 node-id

- Specifies the Drainer node ID. With this ID, this Drainer process can be identified in the cluster.
- Default value: `hostname:port number`. For example, `node-1:8249`.

4.15.5.2.6 data-dir

- Specifies the directory used to store files that need to be saved during Drainer operation.
- Default value: `data.drainer`

4.15.5.2.7 detect-interval

- Specifies the interval (in seconds) at which PD updates the Pump information.
- Default value: `5`

4.15.5.2.8 pd-urls

- The comma-separated list of PD URLs. If multiple addresses are specified, the PD client will automatically attempt to connect to another address if an error occurs when connecting to one address.
- Default value: `http://127.0.0.1:2379`

4.15.5.2.9 initial-commit-ts

- Specifies from which commit timestamp the replication task starts. This configuration is only applicable to the Drainer node that starts replication for the first time. If a checkpoint already exists downstream, the replication will be performed according to the time recorded in the checkpoint.
- Default value: `0`. Drainer will start pulling data from the earliest timestamp of each Pump.

4.15.5.2.10 synced-check-time

- You can access the `/status` path through the HTTP API to query the status of Drainer replication. `synced-check-time` specifies how many minutes from the last successful replication is considered as `synced`, that is, the replication is complete.
- Default value: `5`

4.15.5.2.11 compressor

- Specifies the compression algorithm used for data transfer between Pump and Drainer. Currently only the `gzip` algorithm is supported.
- Default value: "", which means no compression.

4.15.5.2.12 security

This section introduces configuration items related to security.

ssl-ca

- Specifies the file path of the trusted SSL certificate list or CA list. For example, `/path/to/ca.pem`.
- Default value: ""

ssl-cert

- Specifies the path of the X509 certificate file encoded in the PEM format. For example, `/path/to/drainer.pem`.
- Default value: ""

ssl-key

- Specifies the path of the X509 key file encoded in the PEM format. For example, `/path/to/pump-key.pem`.
- Default value: ""

4.15.5.2.13 syncer

The `syncer` section includes configuration items related to the downstream.

db-type

Currently, the following downstream types are supported:

- `mysql`
- `tidb`

- `kafka`
- `file`

Default value: `mysql`

sql-mode

- Specifies the SQL mode when the downstream is the `mysql` or `tidb` type. If there is more than one mode, use commas to separate them.
- Default value: ""

ignore-txn-commit-ts

- Specifies the commit timestamp at which the binlog is ignored, such as `[416815754209656834, 421349811963822081]`.
- Default value: `[]`

ignore-schemas

- Specifies the database to be ignored during replication. If there is more than one database to be ignored, use commas to separate them. If all changes in a binlog file are filtered, the whole binlog file is ignored.
- Default value: `INFORMATION_SCHEMA,PERFORMANCE_SCHEMA,mysql`

ignore-table

Ignores the specified table changes during replication. You can specify multiple tables to be ignored in the `toml` file. For example:

```
[[syncer.ignore-table]]
db-name = "test"
tbl-name = "log"

[[syncer.ignore-table]]
db-name = "test"
tbl-name = "audit"
```

If all changes in a binlog file are filtered, the whole binlog file is ignored.

Default value: `[]`

replicate-do-db

- Specifies the database to be replicated. For example, `[db1, db2]`.
- Default value: `[]`

replicate-do-table

Specifies the table to be replicated. For example:

```
[[syncer.replicate-do-table]]
db-name ="test"
tbl-name = "log"

[[syncer.replicate-do-table]]
db-name ="test"
tbl-name = "~^a.*"
```

Default value: `[]`

txn-batch

- When the downstream is the `mysql` or `tidb` type, DML operations are executed in different batches. This parameter specifies how many DML operations can be included in each transaction.
- Default value: `20`

worker-count

- When the downstream is the `mysql` or `tidb` type, DML operations are executed concurrently. This parameter specifies the concurrency numbers of DML operations.
- Default value: `16`

disable-dispatch

- Disables the concurrency and forcibly set `worker-count` to `1`.
- Default value: `false`

safe-mode

If the safe mode is enabled, Drainer modifies the replication updates in the following way:

- `Insert` is modified to `Replace Into`
- `Update` is modified to `Delete` plus `Replace Into`

Default value: `false`

4.15.5.2.14 syncer.to

The `syncer.to` section introduces different types of downstream configuration items according to configuration types.

mysql/tidb

The following configuration items are related to connection to downstream databases:

- `host`: If this item is not set, TiDB Binlog tries to check the `MYSQL_HOST` environment variable which is `localhost` by default.
- `port`: If this item is not set, TiDB Binlog tries to check the `MYSQL_PORT` environment variable which is `3306` by default.
- `user`: If this item is not set, TiDB Binlog tries to check the `MYSQL_USER` environment variable which is `root` by default.
- `password`: If this item is not set, TiDB Binlog tries to check the `MYSQL_PSWD` environment variable which is `""` by default.

file

- `dir`: Specifies the directory where binlog files are stored. If this item is not set, `data-dir` is used.

kafka

When the downstream is Kafka, the valid configuration items are as follows:

- `zookeeper-addrs`
- `kafka-addrs`
- `kafka-version`
- `kafka-max-messages`
- `topic-name`

4.15.5.2.15 syncer.to.checkpoint

This section introduces a configuration item related to `syncer.to.checkpoint`.

4.15.5.2.16 type

- Specifies in what way the replication progress is saved.
- Available options: `mysql` and `tidb`.
- Default value: The same as the downstream type. For example, when the downstream is `file`, the progress is saved in the local file system; when the downstream is `mysql`, the progress is saved in the downstream database. If you explicitly specify using `mysql` or `tidb` to store the progress, make the following configuration:

- `schema`: `tidb_binlog` by default.

> **Note:**
> When deploying multiple Drainer nodes in the same TiDB cluster, you need to specify a different checkpoint schema for each node. Otherwise, the replication progress of two instances will overwrite each other.

- `host`
- `user`
- `password`
- `port`

4.15.5.3 TiDB Binlog Configuration File

This document introduces the configuration items of TiDB Binlog.

4.15.5.3.1 Pump

This section introduces the configuration items of Pump. For the example of a complete Pump configuration file, see Pump Configuration.

addr

- Specifies the listening address of HTTP API in the format of `host:port`.
- Default value: `127.0.0.1:8250`

advertise-addr

- Specifies the externally accessible HTTP API address. This address is registered in PD in the format of `host:port`.
- Default value: `127.0.0.1:8250`

socket

- The Unix socket address that HTTP API listens to.
- Default value: “”

pd-urls

- Specifies the comma-separated list of PD URLs. If multiple addresses are specified, when the PD client fails to connect to one address, it automatically tries to connect to another address.

- Default value: `http://127.0.0.1:2379`

data-dir

- Specifies the directory where binlogs and their indexes are stored locally.
- Default value: `data.pump`

heartbeat-interval

- Specifies the heartbeat interval (in seconds) at which the latest status is reported to PD.
- Default value: `2`

gen-binlog-interval

- Specifies the interval (in seconds) at which data is written into fake binlog.
- Default value: `3`

gc

- Specifies the number of days (integer) that binlogs can be stored locally. Binlogs stored longer than the specified number of days are automatically deleted.
- Default value: `7`

log-file

- Specifies the path where log files are stored. If the parameter is set to an empty value, log files are not stored.
- Default value: “”

log-level

- Specifies the log level.
- Default value: `info`

node-id

- Specifies the Pump node ID. With this ID, this Pump process can be identified in the cluster.
- Default value: `hostname:port number`. For example, `node-1:8250`.

security

This section introduces configuration items related to security.

ssl-ca

- Specifies the file path of the trusted SSL certificate list or CA list. For example, `/path/to/ca.pem`.
- Default value: “”

ssl-cert

- Specifies the path of the X509 certificate file encoded in the Privacy Enhanced Mail (PEM) format. For example, `/path/to/pump.pem`.
- Default value: “”

ssl-key

- Specifies the path of the X509 key file encoded in the PEM format. For example, `/path/to/pump-key.pem`.
- Default value: “”

storage

This section introduces configuration items related to storage.

sync-log

- Specifies whether to use `fsync` after each **batch** write to binlog to ensure data safety.
- Default value: `true`

kv_chan_cap

- Specifies the number of write requests that the buffer can store before Pump receives these requests.
- Default value: `1048576` (that is, 2 to the power of 20)

slow_write_threshold

- The threshold (in seconds). If it takes longer to write a single binlog file than this specified threshold, the write is considered slow write and `"take a long time to ↪ write binlog"` is output in the log.
- Default value: `1`

stop-write-at-available-space

- Binlog write requests is no longer accepted when the available storage space is below this specified value. You can use the format such as `900 MB`, `5 GB`, and `12 GiB` to specify the storage space. If there is more than one Pump node in the cluster, when a Pump node refuses a write request because of the insufficient space, TiDB will automatically write binlogs to other Pump nodes.
- Default value: `10 GiB`

kv

Currently the storage of Pump is implemented based on GoLevelDB. Under `storage` there is also a `kv` subgroup that is used to adjust the GoLevel configuration. The supported configuration items are shown as below:

- block-cache-capacity
- block-restart-interval
- block-size
- compaction-L0-trigger
- compaction-table-size
- compaction-total-size
- compaction-total-size-multiplier
- write-buffer
- write-L0-pause-trigger
- write-L0-slowdown-trigger

For the detailed description of the above items, see GoLevelDB Document.

4.15.5.3.2 Drainer

This section introduces the configuration items of Drainer. For the example of a complete Drainer configuration file, see Drainer Configuration

addr

- Specifies the listening address of HTTP API in the format of `host:port`.
- Default value: `127.0.0.1:8249`

advertise-addr

- Specifies the externally accessible HTTP API address. This address is registered in PD in the format of `host:port`.
- Default value: `127.0.0.1:8249`

log-file

- Specifies the path where log files are stored. If the parameter is set to an empty value, log files are not stored.
- Default value: “”

log-level

- Specifies the log level.
- Default value: `info`

node-id

- Specifies the Drainer node ID. With this ID, this Drainer process can be identified in the cluster.
- Default value: `hostname:port number`. For example, `node-1:8249`.

data-dir

- Specifies the directory used to store files that need to be saved during Drainer operation.
- Default value: `data.drainer`

detect-interval

- Specifies the interval (in seconds) at which PD updates the Pump information.
- Default value: `5`

pd-urls

- The comma-separated list of PD URLs. If multiple addresses are specified, the PD client will automatically attempt to connect to another address if an error occurs when connecting to one address.
- Default value: `http://127.0.0.1:2379`

initial-commit-ts

- Specifies from which commit timestamp the replication task starts. This configuration is only applicable to the Drainer node that starts replication for the first time. If a checkpoint already exists downstream, the replication will be performed according to the time recorded in the checkpoint.
- Default value: `0`. Drainer will start pulling data from the earliest timestamp of each Pump.

synced-check-time

- You can access the `/status` path through the HTTP API to query the status of Drainer replication. `synced-check-time` specifies how many minutes from the last successful replication is considered as `synced`, that is, the replication is complete.
- Default value: `5`

compressor

- Specifies the compression algorithm used for data transfer between Pump and Drainer. Currently only the `gzip` algorithm is supported.
- Default value: "", which means no compression.

security

This section introduces configuration items related to security.

ssl-ca

- Specifies the file path of the trusted SSL certificate list or CA list. For example, `/path/to/ca.pem`.
- Default value: ""

ssl-cert

- Specifies the path of the X509 certificate file encoded in the PEM format. For example, `/path/to/drainer.pem`.
- Default value: ""

ssl-key

- Specifies the path of the X509 key file encoded in the PEM format. For example, `/path/to/pump-key.pem`.
- Default value: ""

syncer

The `syncer` section includes configuration items related to the downstream.

db-type

Currently, the following downstream types are supported:

- `mysql`
- `tidb`
- `kafka`
- `file`

Default value: `mysql`

sql-mode

- Specifies the SQL mode when the downstream is the `mysql` or `tidb` type. If there is more than one mode, use commas to separate them.
- Default value: ""

ignore-txn-commit-ts

- Specifies the commit timestamp at which the binlog is ignored, such as `[416815754209656834, 421349811963822081]`.
- Default value: `[]`

ignore-schemas

- Specifies the database to be ignored during replication. If there is more than one database to be ignored, use commas to separate them. If all changes in a binlog file are filtered, the whole binlog file is ignored.
- Default value: `INFORMATION_SCHEMA,PERFORMANCE_SCHEMA,mysql`

ignore-table

Ignores the specified table changes during replication. You can specify multiple tables to be ignored in the `toml` file. For example:

```
[[syncer.ignore-table]]
db-name = "test"
tbl-name = "log"

[[syncer.ignore-table]]
db-name = "test"
tbl-name = "audit"
```

If all changes in a binlog file are filtered, the whole binlog file is ignored.

Default value: `[]`

replicate-do-db

- Specifies the database to be replicated. For example, `[db1, db2]`.
- Default value: `[]`

replicate-do-table

Specifies the table to be replicated. For example:

```
[[syncer.replicate-do-table]]
db-name ="test"
tbl-name = "log"

[[syncer.replicate-do-table]]
db-name ="test"
tbl-name = "~^a.*"
```

Default value: `[]`

txn-batch

- When the downstream is the `mysql` or `tidb` type, DML operations are executed in different batches. This parameter specifies how many DML operations can be included in each transaction.
- Default value: `20`

worker-count

- When the downstream is the `mysql` or `tidb` type, DML operations are executed concurrently. This parameter specifies the concurrency numbers of DML operations.
- Default value: `16`

disable-dispatch

- Disables the concurrency and forcibly set `worker-count` to `1`.
- Default value: `false`

safe-mode

If the safe mode is enabled, Drainer modifies the replication updates in the following way:

- `Insert` is modified to `Replace Into`
- `Update` is modified to `Delete` plus `Replace Into`

Default value: `false`

syncer.to

The `syncer.to` section introduces different types of downstream configuration items according to configuration types.

mysql/tidb

The following configuration items are related to connection to downstream databases:

- `host`: If this item is not set, TiDB Binlog tries to check the `MYSQL_HOST` environment variable which is `localhost` by default.
- `port`: If this item is not set, TiDB Binlog tries to check the `MYSQL_PORT` environment variable which is `3306` by default.
- `user`: If this item is not set, TiDB Binlog tries to check the `MYSQL_USER` environment variable which is `root` by default.
- `password`: If this item is not set, TiDB Binlog tries to check the `MYSQL_PSWD` environment variable which is `""` by default.

file

- `dir`: Specifies the directory where binlog files are stored. If this item is not set, `data-dir` is used.

kafka

When the downstream is Kafka, the valid configuration items are as follows:

- `zookeeper-addrs`
- `kafka-addrs`
- `kafka-version`
- `kafka-max-messages`
- `topic-name`

syncer.to.checkpoint

This section introduces a configuration item related to `syncer.to.checkpoint`.

type

- Specifies in what way the replication progress is saved.
- Available options: `mysql` and `tidb`.
- Default value: The same as the downstream type. For example, when the downstream is `file`, the progress is saved in the local file system; when the downstream is `mysql`, the progress is saved in the downstream database. If you explicitly specify using `mysql` or `tidb` to store the progress, make the following configuration:
  - `schema`: `tidb_binlog` by default.

> **Note:**
> When deploying multiple Drainer nodes in the same TiDB cluster, you need to specify a different checkpoint schema for each node. Otherwise, the replication progress of two instances will overwrite each other.

- `host`
- `user`
- `password`
- `port`

4.15.6 Upgrade TiDB Binlog

This document introduces how to upgrade TiDB Binlog that is deployed with TiDB Ansible and deployed manually to the latest cluster version. There is also a section on how to upgrade TiDB Binlog from an earlier incompatible version (Kafka/Local version) to the latest version.

4.15.6.1 Upgrade TiDB Binlog deployed with TiDB Ansible

Follow the steps in this section if you deploy TiDB Binlog with TiDB Ansible Playbook.

4.15.6.1.1 Upgrade Pump

First, upgrade the Pump component:

1. Copy the new version of the binary `pump` file into the`({ resources_dir })/bin` directory.
2. Execute the `ansible-playbook rolling_update.yml --tags=pump` command to perform a rolling update for Pump.

4.15.6.1.2 Upgrade Drainer

Second, upgrade the Drainer component:

1. Copy the new version of the binary `drainer` file into the`({ resources_dir })/bin` directory.
2. Execute the `ansible-playbook stop_drainer.yml --tags=drainer` command.
3. Execute the `ansible-playbook start_drainer.yml --tags=drainer` command.

4.15.6.2 Upgrade TiDB Binlog deployed manually

Follow the steps in this section if you deploy TiDB Binlog manually.

4.15.6.2.1 Upgrade Pump

First, upgrade each Pump instance in the cluster one by one. This ensures that there are always Pump instances in the cluster that can receive binlogs from TiDB. The steps are as below:

1. Replace the original file with the new version of `pump`.
2. Restart the Pump process.

4.15.6.2.2 Upgrade Drainer

Second, upgrade the Drainer component:

1. Replace the original file with the new version of `drainer`.
2. Restart the Drainer process.

4.15.6.3 Upgrade TiDB Binlog from Kafka/Local version to the cluster version

The new TiDB versions (v2.0.8-binlog, v2.1.0-rc.5 or later) are not compatible with the Kafka version or Local version of TiDB Binlog. If TiDB is upgraded to one of the new versions, it is required to use the cluster version of TiDB Binlog. If the Kafka or local version of TiDB Binlog is used before upgrading, you need to upgrade your TiDB Binlog to the cluster version.

The corresponding relationship between TiDB Binlog versions and TiDB versions is shown in the following table:

TiDB Binlog version	TiDB version	Note
Local	TiDB 1.0 or earlier	
Kafka	TiDB 1.0 ~ TiDB 2.1 RC5	TiDB 1.0 supports both the local and Kafka versions of TiDB Binlog.
Cluster	TiDB v2.0.8-binlog, TiDB 2.1 RC5 or later	TiDB v2.0.8-binlog is a special 2.0 version supporting the cluster version of TiDB Binlog.

4.15.6.3.1 Upgrade process

> **Note:**
>
> If importing the full data is acceptable, you can abandon the old version and deploy TiDB Binlog following TiDB Binlog Cluster Deployment.

If you want to resume replication from the original checkpoint, perform the following steps to upgrade TiDB Binlog:

1. Deploy the new version of Pump.
2. Stop the TiDB cluster service.

3. Upgrade TiDB and the configuration, and write the binlog data to the new Pump cluster.

4. Reconnect the TiDB cluster to the service.

5. Make sure that the old version of Drainer has replicated the data in the old version of Pump to the downstream completely;

   Query the `status` interface of Drainer, command as below:

```
curl 'http://172.16.10.49:8249/status'
```

```
{"PumpPos":{"172.16.10.49:8250":{"offset":32686}},"Synced": true ,"
    ↪ DepositWindow":{"Upper":398907800202772481,"Lower
    ↪ ":398907799455662081}}
```

   If the return value of `Synced` is True, it means Drainer has replicated the data in the old version of Pump to the downstream completely.

6. Start the new version of Drainer.

7. Close the Pump and Drainer of the old versions and the dependent Kafka and ZooKeeper.

4.15.7 Reparo User Guide

Reparo is a TiDB Binlog tool, used to recover the incremental data. To back up the incremental data, you can use Drainer of TiDB Binlog to output the binlog data in the protobuf format to files. To restore the incremental data, you can use Reparo to parse the binlog data in the files and apply the binlog in TiDB/MySQL.

Download Reparo via tidb-binlog-cluster-latest-linux-amd64.tar.gz

4.15.7.1 Reparo usage

4.15.7.1.1 Description of command line parameters

```
Usage of Reparo:
-L string
    The level of the output information of logs
    Value: "debug"/"info"/"warn"/"error"/"fatal" ("info" by default)
-V Prints the version.
-c int
    The number of concurrencies in the downstream for the replication
        ↪ process (`16` by default). A higher value indicates a better
        ↪ throughput for the replication.
-config string
```

```
    The path of the configuration file
    If the configuration file is specified, Reparo reads the configuration
       ↪ data in this file.
    If the configuration data also exists in the command line parameters,
       ↪ Reparo uses the configuration data in the command line parameters
       ↪ to cover that in the configuration file.
-data-dir string
    The storage directory for the binlog file in the protobuf format that
       ↪ Drainer outputs ("data.drainer" by default)
-dest-type string
    The downstream service type
    Value: "print"/"mysql" ("print" by default)
    If it is set to "print", the data is parsed and printed to standard
       ↪ output while the SQL statement is not executed.
    If it is set to "mysql", you need to configure the "host", "port", "user
       ↪ " and "password" information in the configuration file.
-log-file string
    The path of the log file
-log-rotate string
    The switch frequency of log files
    Value: "hour"/"day"
-start-datetime string
    Specifies the time point for starting recovery.
    Format: "2006-01-02 15:04:05"
    If it is not set, the recovery process starts from the earliest binlog
       ↪ file.
-stop-datetime string
    Specifies the time point of finishing the recovery process.
    Format: "2006-01-02 15:04:05"
    If it is not set, the recovery process ends up with the last binlog file
       ↪ .
-safe-mode bool
    Specifies whether to enable safe mode. When enabled, it supports
       ↪ repeated replication.
-txn-batch int
    The number of SQL statements in a transaction that is output to the
       ↪ downstream database (`20` by default).
```

4.15.7.1.2 Description of the configuration file

```
### The storage directory for the binlog file in the protobuf format that
    ↪ Drainer outputs
data-dir = "./data.drainer"
```

```
### The level of the output information of logs
### Value: "debug"/"info"/"warn"/"error"/"fatal" ("info" by default)
log-level = "info"

### Uses `start-datetime` and `stop-datetime` to specify the time range in
    ↪ which
### the binlog files are to be recovered.
### Format: "2006-01-02 15:04:05"
### start-datetime = ""
### stop-datetime = ""

### Correspond to `start-datetime` and `stop-datetime` respectively.
### They are used to specify the time range in which the binlog files are to
    ↪  be recovered.
### If `start-datetime` and `stop-datetime` are set, there is no need to set
    ↪  `start-tso` and `stop-tso`.
### start-tso = 0
### stop-tso = 0

### The downstream service type
### Value: "print"/"mysql" ("print" by default)
### If it is set to "print", the data is parsed and printed to standard
    ↪ output
### while the SQL statement is not executed.
### If it is set to "mysql", you need to configure `host`, `port`, `user`
    ↪ and `password` in [dest-db].
dest-type = "mysql"

### The number of SQL statements in a transaction that is output to the
    ↪ downstream database (`20` by default).
txn-batch = 20

### The number of concurrencies in the downstream for the replication
    ↪ process (`16` by default). A higher value indicates a better
    ↪ throughput for the replication.
worker-count = 16

### Safe-mode configuration
### Value: "true"/"false" ("false" by default)
### If it is set to "true", Reparo splits the `UPDATE` statement into a `
    ↪ DELETE` statement plus a `REPLACE` statement.
safe-mode = false

### `replicate-do-db` and `replicate-do-table` specify the database and
    ↪ table to be recovered.
```

```
### `replicate-do-db` has priority over `replicate-do-table`.
### You can use a regular expression for configuration. The regular
    ↪ expression should start with "~".
### The configuration method for `replicate-do-db` and `replicate-do-table`
    ↪ is
### the same with that for `replicate-do-db` and `replicate-do-table` of
    ↪ Drainer.
### replicate-do-db = ["~^b.*","s1"]
### [[replicate-do-table]]
### db-name ="test"
### tbl-name = "log"
### [[replicate-do-table]]
### db-name ="test"
### tbl-name = "~^a.*"

### If `dest-type` is set to `mysql`, `dest-db` needs to be configured.
[dest-db]
host = "127.0.0.1"
port = 3309
user = "root"
password = ""
```

4.15.7.1.3 Start example

```
./bin/reparo -config reparo.toml
```

Note:

- `data-dir` specifies the directory for the binlog file that Drainer outputs.
- Both `start-datatime` and `start-tso` are used to specify the time point for starting recovery, but they are different in the time format. If they are not set, the recovery process starts from the earliest binlog file by default.
- Both `stop-datetime` and `stop-tso` are used to specify the time point for finishing recovery, but they are different in the time format. If they are not set, the recovery process ends up with the last binlog file by default.
- `dest-type` specifies the destination type. Its value can be "mysql" and "print."

  - When it is set to `mysql`, the data can be recovered to MySQL or TiDB that uses or is compatible with the MySQL protocol. In this case, you need to specify the database information in `[dest-db]` of the configuration information.
  - When it is set to `print`, only the binlog information is printed. It is generally used for debugging and checking the binlog information. In this case, there is no need to specify `[dest-db]`.
- `replicate-do-db` specifies the database for recovery. If it is not set, all the databases are to be recovered.
- `replicate-do-table` specifies the table for recovery. If it is not set, all the tables are to be recovered.

4.15.8 Binlog Consumer Client User Guide

Binlog Consumer Client is used to consume TiDB secondary binlog data from Kafka and output the data in a specific format. Currently, Drainer supports multiple kinds of down streaming, including MySQL, TiDB, file and Kafka. But sometimes users have customized requirements for outputting data to other formats, for example, Elasticsearch and Hive, so this feature is introduced.

4.15.8.1 Configure Drainer

Modify the configuration file of Drainer and set it to output the data to Kafka:

```
[syncer]
db-type = "kafka"

[syncer.to]
### the Kafka address
kafka-addrs = "127.0.0.1:9092"
### the Kafka version
kafka-version = "0.8.2.0"
```

4.15.8.2 Customized development

4.15.8.2.1 Data format

Firstly, you need to obtain the format information of the data which is output to Kafka by Drainer:

```
// `Column` stores the column data in the corresponding variable based on
   ↪ the data type.
```

```
message Column {
  // Indicates whether the data is null
  optional bool is_null = 1 [ default = false ];
  // Stores `int` data
  optional int64 int64_value = 2;
  // Stores `uint`, `enum`, and `set` data
  optional uint64 uint64_value = 3;
  // Stores `float` and `double` data
  optional double double_value = 4;
  // Stores `bit`, `blob`, `binary` and `json` data
  optional bytes bytes_value = 5;
  // Stores `date`, `time`, `decimal`, `text`, `char` data
  optional string string_value = 6;
}

// `ColumnInfo` stores the column information, including the column name,
    ↪ type, and whether it is the primary key.
message ColumnInfo {
  optional string name = 1 [ (gogoproto.nullable) = false ];
  // the lower case column field type in MySQL
  // https://dev.mysql.com/doc/refman/8.0/en/data-types.html
  // for the `numeric` type: int bigint smallint tinyint float double
     ↪ decimal bit
  // for the `string` type: text longtext mediumtext char tinytext varchar
  // blob longblob mediumblob binary tinyblob varbinary
  // enum set
  // for the `json` type: json
  optional string mysql_type = 2 [ (gogoproto.nullable) = false ];
  optional bool is_primary_key = 3 [ (gogoproto.nullable) = false ];
}

// `Row` stores the actual data of a row.
message Row { repeated Column columns = 1; }

// `MutationType` indicates the DML type.
enum MutationType {
  Insert = 0;
  Update = 1;
  Delete = 2;
}

// `Table` contains mutations in a table.
message Table {
  optional string schema_name = 1;
  optional string table_name = 2;
```

```
  repeated ColumnInfo column_info = 3;
  repeated TableMutation mutations = 4;
}

// `TableMutation` stores mutations of a row.
message TableMutation {
  required MutationType type = 1;
  // data after modification
  required Row row = 2;
  // data before modification. It only takes effect for `Update MutationType
     ↪ `.
  optional Row change_row = 3;
}

// `DMLData` stores all the mutations caused by DML in a transaction.
message DMLData {
  // `tables` contains all the table changes in the transaction.
  repeated Table tables = 1;
}

// `DDLData` stores the DDL information.
message DDLData {
  // the database used currently
  optional string schema_name = 1;
  // the relates table
  optional string table_name = 2;
  // `ddl_query` is the original DDL statement query.
  optional bytes ddl_query = 3;
}

// `BinlogType` indicates the binlog type, including DML and DDL.
enum BinlogType {
  DML = 0; // Has `dml_data`
  DDL = 1; // Has `ddl_query`
}

// `Binlog` stores all the changes in a transaction. Kafka stores the
   ↪ serialized result of the structure data.
message Binlog {
  optional BinlogType type = 1 [ (gogoproto.nullable) = false ];
  optional int64 commit_ts = 2 [ (gogoproto.nullable) = false ];
  optional DMLData dml_data = 3;
  optional DDLData ddl_data = 4;
}
```

For the definition of the data format, see `binlog.proto`.

4.15.8.2.2 Driver

The TiDB-Tools project provides Driver, which is used to read the binlog data in Kafka. It has the following features:

- Read the Kafka data.
- Locate the binlog stored in Kafka based on `commit ts`.

You need to configure the following information when using Driver:

- `KafkaAddr`: the address of the Kafka cluster
- `CommitTS`: from which `commit ts` to start reading the binlog
- `Offset`: from which Kafka `offset` to start reading data. If `CommitTS` is set, you needn't configure this parameter.
- `ClusterID`: the cluster ID of the TiDB cluster
- `Topic`: the topic name of Kafka. If Topic is empty, use the default name in Drainer `<ClusterID>_obinlog`.

You can use Driver by quoting the Driver code in package and refer to the example code provided by Driver to learn how to use Driver and parse the binlog data.

Currently, two examples are provided:

- Using Driver to replicate data to MySQL. This example shows how to convert a binlog to SQL
- Using Driver to print data

Note:

- The example code only shows how to use Driver. If you want to use Driver in the production environment, you need to optimize the code.
- Currently, only the Golang version of Driver and example code are available. If you want to use other languages, you need to generate the code file in the corresponding language based on the binlog proto file and develop an application to read the binlog data in Kafka, parse the data, and output the data to the downstream. You are also welcome to optimize the example code and submit the example code of other languages to TiDB-Tools.

4.15.9 TiDB Binlog Relay Log

When replicating binlogs, Drainer splits transactions from the upstream and replicates the split transactions concurrently to the downstream.

In extreme cases where the upstream clusters are not available and Drainer exits abnormally, the downstream clusters (MySQL or TiDB) might be in the intermediate states with inconsistent data. In such cases, Drainer can use the relay log to ensure that the downstream clusters are in a consistent state.

4.15.9.1 Consistent state during Drainer replication

The downstream clusters reaching a consistent state means the data of the downstream clusters are the same as the snapshot of the upstream which sets `tidb_snapshot = ts`.

The checkpoint consistency means Drainer checkpoint saves the consistent state of replication in `consistent`. When Drainer runs, `consistent` is `false`. After Drainer exits normally, `consistent` is set to `true`.

You can query the downstream checkpoint table as follows:

```
select * from tidb_binlog.checkpoint;
```

```
+--------------------+-------------------------------------------------------------------+
   ↪
| clusterID          | checkPoint
   ↪                                                  |
+--------------------+-------------------------------------------------------------------+
   ↪
| 6791641053252586769 | {"consistent":false,"commitTS":414529105591271429,"
   ↪ ts-map":{}} |
+--------------------+-------------------------------------------------------------------+
   ↪
```

4.15.9.2 Implementation principles

After Drainer enables the relay log, it first writes the binlog events to the disks and then replicates the events to the downstream clusters.

If the upstream clusters are not available, Drainer can restore the downstream clusters to a consistent state by reading the relay log.

> **Note:**
>
> If the relay log data is lost at the same time, this method does not work, but its incidence is very low. In addition, you can use the Network File System to ensure data safety of the relay log.

4.15.9.2.1 Trigger scenarios where Drainer consumes binlogs from the relay log

When Drainer is started, if it fails to connect to the Placement Driver (PD) of the upstream clusters, and it detects that `consistent = false` in the checkpoint, Drainer will try to read the relay log, and restore the downstream clusters to a consistent state. After that, the Drainer process sets the checkpoint `consistent` to `true` and then exits.

4.15.9.2.2 GC mechanism of relay log

Before data is replicated to the downstream, Drainer writes data to the relay log file. If the size of a relay log file reaches 10 MB (by default) and the binlog data of the current transaction is completely written, Drainer starts to write data to the next relay log file. After Drainer successfully replicates data to the downstream, it automatically cleans up the relay log files whose data has been replicated. The relay log into which data is currently being written will not be cleaned up.

4.15.9.3 Configuration

To enable the relay log, add the following configuration in Drainer:

```
[syncer.relay]
### It saves the directory of the relay log. The relay log is not enabled if
    ↪ the value is empty.
### The configuration only comes to effect if the downstream is TiDB or
    ↪ MySQL.
log-dir = "/dir/to/save/log"
### The size limit of a single relay log file (unit: byte).
### When the size of a relay log file reaches this limit, data is written to
    ↪ the next relay log file.
max-file-size = 10485760
```

4.15.10 Bidirectional Replication between TiDB Clusters

This document describes the bidirectional replication between two TiDB clusters, how the replication works, how to enable it, and how to replicate DDL operations.

4.15.10.1 User scenario

If you want two TiDB clusters to exchange data changes with each other, TiDB Binlog allows you to do that. For example, you want cluster A and cluster B to replicate data with each other.

Note:

The data written to these two clusters must be conflict-free, that is, in the two clusters, the same primary key or the rows with the unique index of the tables must not be modified.

The user scenario is shown as below:

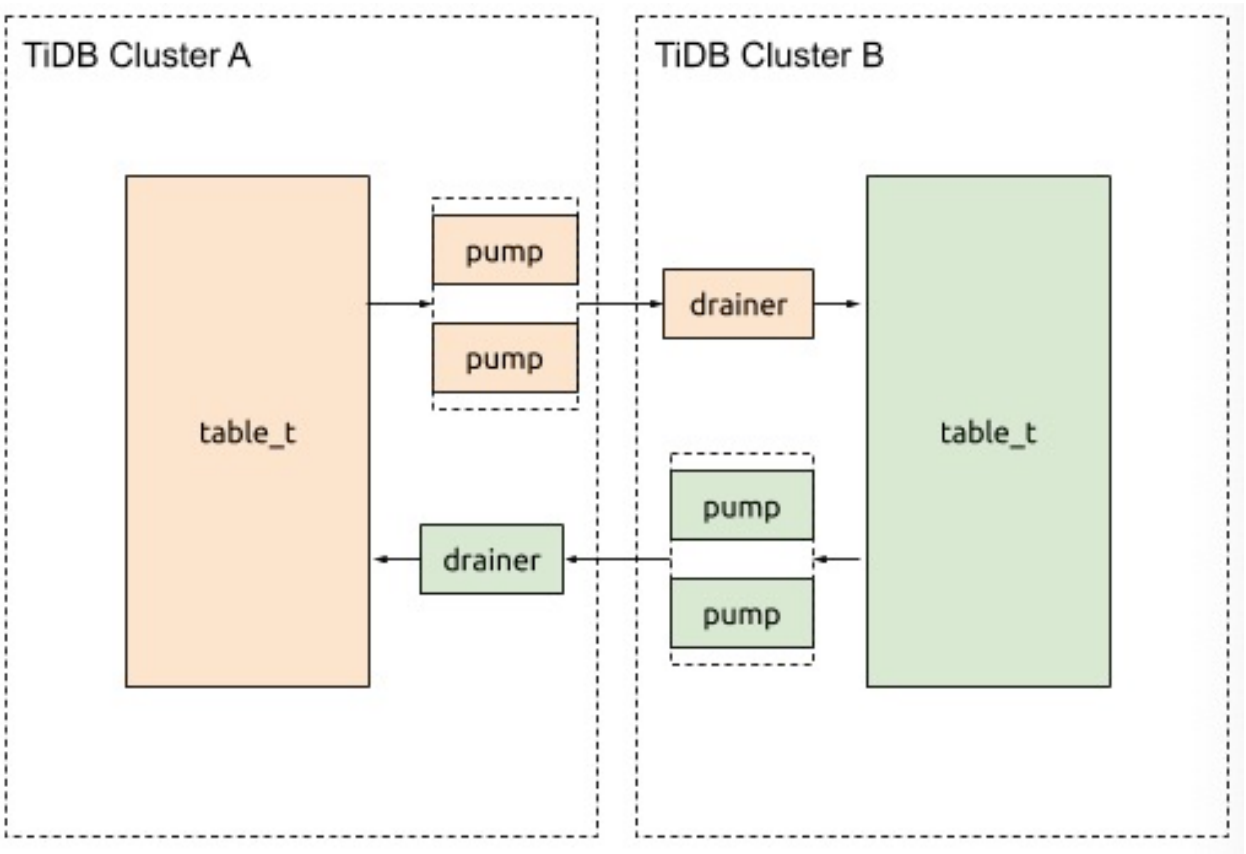

Figure 330: Architect

4.15.10.2 Implementation details

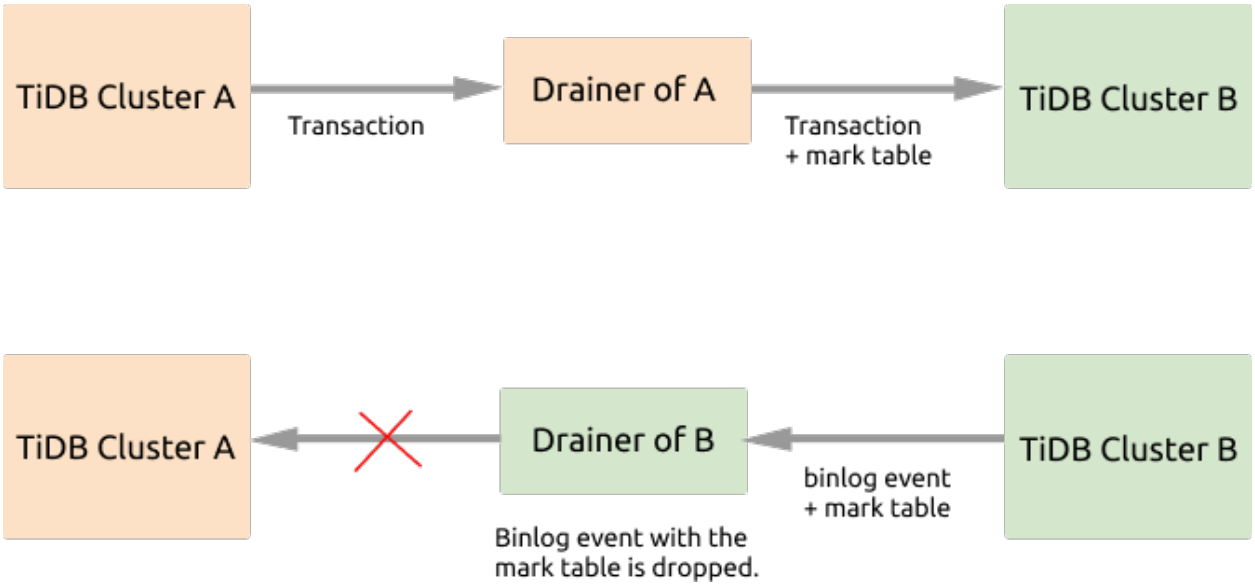

Figure 331: Mark Table

If the bidirectional replication is enabled between cluster A and cluster B, the data written to cluster A will be replicated to cluster B, and then these data changes will be replicated back to cluster A, which causes an infinite loop of replication. From the figure above, you can see that during the data replication, Drainer marks the binlog events, and filters out the marked events to avoid such a replication loop.

The detailed implementation is described as follows:

1. Start the TiDB Binlog replication program for each of the two clusters.
2. When the transaction to be replicated passes through the Drainer of cluster A, this Drainer adds the `_drainer_repl_mark` table to the transaction, writes this DML event update to the mark table, and replicate this transaction to cluster B.
3. Cluster B returns binlog events with the `_drainer_repl_mark` mark table to cluster A. The Drainer of cluster B identifies the mark table with the DML event when parsing the binlog event, and gives up replicating this binlog event to cluster A.

The replication process from cluster B to cluster A is the same as above. The two clusters can be upstream and downstream of each other.

Note:

- When updating the `_drainer_repl_mark` mark table, data changes are required to generate binlogs.

- DDL operations are not transactional, so you need to use the one-way replication method to replicate DDL operations. See Replicate DDL operations for details.

Drainer can use a unique ID for each connection to downstream to avoid conflicts. `channel_id` is used to indicate a channel for bidirectional replication. The two clusters should have the same `channel_id` configuration (with the same value).

If you add or delete columns in the upstream, there might be extra or missing columns of the data to be replicated to the downstream. Drainer allows this situation by ignoring the extra columns or by inserting default values to the missing columns.

4.15.10.3 Mark table

The `_drainer_repl_mark` mark table has the following structure:

```
CREATE TABLE `_drainer_repl_mark` (
  `id` bigint(20) NOT NULL,
  `channel_id` bigint(20) NOT NULL DEFAULT '0',
  `val` bigint(20) DEFAULT '0',
  `channel_info` varchar(64) DEFAULT NULL,
  PRIMARY KEY (`id`,`channel_id`)
);
```

Drainer uses the following SQL statement to update `_drainer_repl_mark`, which ensures data change and the generation of binlog:

```
update drainer_repl_mark set val = val + 1 where id = ? && channel_id = ?;
```

4.15.10.4 Replicate DDL operations

Because Drainer cannot add the mark table to DDL operations, you can only use the one-way replication method to replicate DDL operations.

For example, if DDL replication is enabled from cluster A to cluster B, then the replication is disabled from cluster B to cluster A. This means that all DDL operations are performed on cluster A.

Note:

DDL operations cannot be executed on two clusters at the same time. When a DDL operation is executed, if any DML operation is being executed at the same time or any DML binlog is being replicated, the upstream and downstream table structures of the DML replication might be inconsistent.

4.15.10.5 Configure and enable bidirectional replication

For bidirectional replication between cluster A and cluster B, assume that all DDL operations are executed on cluster A. On the replication path from cluster A to cluster B, add the following configuration to Drainer:

```
[syncer]
loopback-control = true
channel-id = 1 # Configures the same ID for both clusters to be replicated.
sync-ddl = true # Enables it if you need to perform DDL replication.

[syncer.to]
### 1 means SyncFullColumn and 2 means SyncPartialColumn.
### If set to SyncPartialColumn, Drainer allows the downstream table
### structure to have more or fewer columns than the data to be replicated
### And remove the STRICT_TRANS_TABLES of the SQL mode to allow fewer
   ↪ columns, and insert zero values to the downstream.
sync-mode = 2

### Ignores the checkpoint table.
[[syncer.ignore-table]]
db-name = "tidb_binlog"
tbl-name = "checkpoint"
```

On the replication path from cluster B to cluster A, add the following configuration to Drainer:

```
[syncer]
loopback-control = true
channel-id = 1 # Configures the same ID for both clusters to be replicated.
sync-ddl = false # Disables it if you do not need to perform DDL replication
   ↪ .

[syncer.to]
### 1 means SyncFullColumn and 2 means SyncPartialColumn.
### If set to SyncPartialColumn, Drainer allows the downstream table
### structure to have more or fewer columns than the data to be replicated
### And remove the STRICT_TRANS_TABLES of the SQL mode to allow fewer
   ↪ columns, and insert zero values to the downstream.
sync-mode = 2

### Ignores the checkpoint table.
[[syncer.ignore-table]]
db-name = "tidb_binlog"
tbl-name = "checkpoint"
```

4.15.11 TiDB Binlog Glossary

This document lists the terms used in the logs, monitoring, configurations, and documentation of TiDB Binlog.

4.15.11.1 Binlog

In TiDB Binlog, binlogs refer to the binary log data from TiDB. They also refer to the binary log data that Drainer writes to Kafka or files. The former and the latter are in different formats. In addition, binlogs in TiDB and binlogs in MySQL are also in different formats.

4.15.11.2 Binlog event

The DML binlogs from TiDB have three types of event: `INSERT`, `UPDATE`, and `DELETE`. In the monitoring dashboard of Drainer, you can see the number of different events that correspond to the replication data.

4.15.11.3 Checkpoint

A checkpoint indicates the position from which a replication task is paused and resumed, or is stopped and restarted. It records the commit-ts that Drainer replicates to the downstream. When restarted, Drainer reads the checkpoint and starts replicating data from the corresponding commit-ts.

4.15.11.4 Safe mode

Safe mode refers to the mode that supports the idempotent import of DML when a primary key or unique index exists in the table schema in the incremental replication task.

In this mode, the `INSERT` statement is re-written as `REPLACE`, and the `UPDATE` statement is re-written as `DELETE` and `REPLACE`. Then the re-written statement is executed to the downstream. Safe mode is automatically enabled within 5 minutes after Drainer is started. You can manually enable the mode by modifying the `safe-mode` parameter in the configuration file, but this configuration is valid only when the downstream is MySQL or TiDB.

4.15.12 Troubleshoot

4.15.12.1 TiDB Binlog Troubleshooting

This document describes how to troubleshoot TiDB Binlog to find the problem.

If you encounter errors while running TiDB Binlog, take the following steps to troubleshoot:

1. Check whether each monitoring metric is normal or not. Refer to TiDB Binlog Monitoring for details.

2. Use the binlogctl tool to check whether the state of each Pump or Drainer node is normal or not.

3. Check whether `ERROR` or `WARN` exists in the Pump log or Drainer log.

After finding out the problem by the above steps, refer to FAQ and TiDB Binlog Error Handling for the solution. If you fail to find the solution or the solution provided does not help, submit an issue for help.

4.15.12.2 TiDB Binlog Error Handling

This document introduces common errors that you might encounter and solutions to these errors when you use TiDB Binlog.

4.15.12.2.1 `kafka server: Message was too large, server rejected it to avoid allocation error` is returned when Drainer replicates data to Kafka

Cause: Executing a large transaction in TiDB generates binlog data of a large size, which might exceed Kafka's limit on the message size.

Solution: Adjust the configuration parameters of Kafka as shown below:

```
message.max.bytes=1073741824
replica.fetch.max.bytes=1073741824
fetch.message.max.bytes=1073741824
```

4.15.12.2.2 Pump returns `no space left on device` error

Cause: The local disk space is insufficient for Pump to write binlog data normally.

Solution: Clean up the disk space and then restart Pump.

4.15.12.2.3 `fail to notify all living drainer` is returned when Pump is started

Cause: When Pump is started, it notifies all Drainer nodes that are in the `online` state. If it fails to notify Drainer, this error log is printed.

Solution: Use the binlogctl tool to check whether each Drainer node is normal or not. This is to ensure that all Drainer nodes that are in the `online` state are working normally. If the state of a Drainer node is not consistent with its actual working status, use the binlogctl tool to change its state and then restart Pump.

4.15.13 TiDB Binlog FAQ

This document collects the frequently asked questions (FAQs) about TiDB Binlog.

4.15.13.1 What is the impact of enabling TiDB Binlog on the performance of TiDB?

- There is no impact on the query.
- There is a slight performance impact on `INSERT`, `DELETE` and `UPDATE` transactions. In latency, a p-binlog is written concurrently in the TiKV prewrite stage before the transactions are committed. Generally, writing binlog is faster than TiKV prewrite, so it does not increase latency. You can check the response time of writing binlog in Pump's monitoring panel.

4.15.13.2 How high is the replication latency of TiDB Binlog?

The latency of TiDB Binlog replication is measured in seconds, which is generally about 3 seconds during off-peak hours.

4.15.13.3 What privileges does Drainer need to replicate data to the downstream MySQL or TiDB cluster?

To replicate data to the downstream MySQL or TiDB cluster, Drainer must have the following privileges:

- Insert
- Update
- Delete
- Create
- Drop
- Alter
- Execute
- Index
- Select

4.15.13.4 What can I do if the Pump disk is almost full?

1. Check whether Pump's GC works well:
    - Check whether the **gc_tso** time in Pump's monitoring panel is identical with that of the configuration file. For Pump of version <= v3.0.0, GC works after the non-offline drainer consumes data. If there are drainer instances that are no longer in use, use binlogctl to make them offline.
2. If GC works well, perform the following steps to reduce the amount of space required for a single Pump:
    - Modify the **GC** parameter of Pump to reduce the number of days to retain data.
    - Add pump instances.

4.15.13.5 What can I do if Drainer replication is interrupted?

Execute the following command to check whether the status of Pump is normal and whether all the Pump instances that are not in the `offline` state are running.

```
binlogctl -cmd pumps
```

Then, check whether the Drainer monitor or log outputs corresponding errors. If so, resolve them accordingly.

4.15.13.6 What can I do if Drainer is slow to replicate data to the downstream MySQL or TiDB cluster?

Check the following monitoring items:

- For the **Drainer Event** monitoring metric, check the speed of Drainer replicating `INSERT`, `UPDATE` and `DELETE` transactions to the downstream per second.
- For the **SQL Query Time** monitoring metric, check the time Drainer takes to execute SQL statements in the downstream.

Possible causes and solutions for slow replication:

- If the replicated database contains a table without a primary key or unique index, add a primary key to the table.
- If the latency between Drainer and the downstream is high, increase the value of the `worker-count` parameter of Drainer. For cross-datacenter replication, it is recommended to deploy Drainer in the downstream.
- If the load in the downstream is not high, increase the value of the `worker-count` parameter of Drainer.

4.15.13.7 What can I do if a Pump instance crashes?

If a Pump instance crashes, Drainer cannot replicate data to the downstream because it cannot obtain the data of this instance. If this Pump instance can recover to the normal state, Drainer resumes replication; if not, perform the following steps:

1. Use binlogctl to change the state of this Pump instance to `offline` to discard the data of this Pump instance.
2. Because Drainer cannot obtain the data of this pump instance, the data in the downstream and upstream is inconsistent. In this situation, perform full and incremental backups again. The steps are as follows:

    1. Stop the Drainer.

2. Perform a full backup in the upstream.
3. Clear the data in the downstream including the `tidb_binlog.checkpoint` table.
4. Restore the full backup to the downstream.
5. Deploy Drainer and use `initialCommitTs` (set `initialCommitTs` as the snapshot timestamp of the full backup) as the start point of initial replication.

4.15.13.8 What is checkpoint?

Checkpoint records the `commit-ts` that Drainer replicates to the downstream. When Drainer restarts, it reads the checkpoint and then replicates data to the downstream starting from the corresponding `commit-ts`. The `["write save point"] [ts =411222863322546177]` Drainer log means saving the checkpoint with the corresponding timestamp.

Checkpoint is saved in different ways for different types of downstream platforms:

- For MySQL/TiDB, it is saved in the `tidb_binlog.checkpoint` table.
- For Kafka/file, it is saved in the file of the corresponding configuration directory.

The data of kafka/file contains `commit-ts`, so if the checkpoint is lost, you can check the latest `commit-ts` of the downstream data by consuming the latest data in the downstream .

Drainer reads the checkpoint when it starts. If Drainer cannot read the checkpoint, it uses the configured `initialCommitTs` as the start point of the initial replication.

4.15.13.9 How to redeploy Drainer on the new machine when Drainer fails and the data in the downstream remains?

If the data in the downstream is not affected, you can redeploy Drainer on the new machine as long as the data can be replicated from the corresponding checkpoint.

- If the checkpoint is not lost, perform the following steps:
    1. Deploy and start a new Drainer (Drainer can read checkpoint and resumes replication).
    2. Use binlogctl to change the state of the old Drainer to `offline`.
- If the checkpoint is lost, perform the following steps:
    1. To deploy a new Drainer, obtain the `commit-ts` of the old Drainer as the `initialCommitTs` of the new Drainer.
    2. Use binlogctl to change the state of the old Drainer to `offline`.

4.15.13.10 How to restore the data of a cluster using a full backup and a binlog backup file?

1. Clean up the cluster and restore a full backup.
2. To restore the latest data of the backup file, use Reparo to set `start-tso` = {snapshot timestamp of the full backup + 1} and `end-ts` = 0 (or you can specify a point in time).

4.15.13.11 How to redeploy Drainer when enabling `ignore-error` in Primary-Secondary replication triggers a critical error?

If a critical error is triggered when TiDB fails to write binlog after enabling `ignore-error`, TiDB stops writing binlog and binlog data loss occurs. To resume replication, perform the following steps:

1. Stop the Drainer instance.
2. Restart the `tidb-server` instance that triggers critical error and resume writing binlog (TiDB does not write binlog to Pump after critical error is triggered).
3. Perform a full backup in the upstream.
4. Clear the data in the downstream including the `tidb_binlog.checkpoint` table.
5. Restore the full backup to the downstream.
6. Deploy Drainer and use `initialCommitTs` (set `initialCommitTs` as the snapshot timestamp of the full backup) as the start point of initial replication.

4.15.13.12 When can I pause or close a Pump or Drainer node?

Refer to TiDB Binlog Cluster Operations to learn the description of the Pump or Drainer state and how to start and exit the process.

Pause a Pump or Drainer node when you need to temporarily stop the service. For example:

- Version upgrade

Use the new binary to restart the service after the process is stopped.

- Server maintenance

When the server needs a downtime maintenance, exit the process and restart the service after the maintenance is finished.

Close a Pump or Drainer node when you no longer need the service. For example:

- Pump scale-in

If you do not need too many Pump services, close some of them.

- Cancelling replication tasks

If you no longer need to replicate data to a downstream database, close the corresponding Drainer node.

- Service migration

If you need to migrate the service to another server, close the service and re-deploy it on the new server.

4.15.13.13 How can I pause a Pump or Drainer process?

- Directly kill the process.

> **Note:**
> Do not use the `kill -9` command. Otherwise, the Pump or Drainer node cannot process signals.

- If the Pump or Drainer node runs in the foreground, pause it by pressing Ctrl+C.
- Use the `pause-pump` or `pause-drainer` command in binlogctl.

4.15.13.14 Can I use the `update-pump` or `update-drainer` command in binlogctl to pause the Pump or Drainer service?

No. The `update-pump` or `update-drainer` command directly modifies the state information saved in PD without notifying Pump or Drainer to perform the corresponding operation. Misusing the two commands can interrupt data replication and might even cause data loss.

4.15.13.15 Can I use the `update-pump` or `update-drainer` command in binlogctl to close the Pump or Drainer service?

No. The `update-pump` or `update-drainer` command directly modifies the state information saved in PD without notifying Pump or Drainer to perform the corresponding operation. Misusing the two commands interrupts data replication and might even cause data inconsistency. For example:

- When a Pump node runs normally or is in the `paused` state, if you use the `update-pump` command to set the Pump state to `offline`, the Drainer node stops pulling the binlog data from the `offline` Pump. In this situation, the newest binlog cannot be replicated to the Drainer node, causing data inconsistency between upstream and downstream.
- When a Drainer node runs normally, if you use the `update-drainer` command to set the Drainer state to `offline`, the newly started Pump node only notifies Drainer nodes in the `online` state. In this situation, the `offline` Drainer fails to pull the binlog data from the Pump node in time, causing data inconsistency between upstream and downstream.

4.15.13.16 When can I use the `update-pump` command in binlogctl to set the Pump state to `paused`?

In some abnormal situations, Pump fails to correctly maintain its state. Then, use the `update-pump` command to modify the state.

For example, when a Pump process is exited abnormally (caused by directly exiting the process when a panic occurs or mistakenly using the `kill -9` command to kill the process), the Pump state information saved in PD is still `online`. In this situation, if you do not need to restart Pump to recover the service at the moment, use the `update-pump` command to update the Pump state to `paused`. Then, interruptions can be avoided when TiDB writes binlogs and Drainer pulls binlogs.

4.15.13.17 When can I use the `update-drainer` command in binlogctl to set the Drainer state to `paused`?

In some abnormal situations, the Drainer node fails to correctly maintain its state, which has influenced the replication task. Then, use the `update-drainer` command to modify the state.

For example, when a Drainer process is exited abnormally (caused by directly exiting the process when a panic occurs or mistakenly using the `kill -9` command to kill the process), the Drainer state information saved in PD is still `online`. When a Pump node is started, it fails to notify the exited Drainer node (the `notify drainer ...` error), which cause the Pump node failure. In this situation, use the `update-drainer` command to update the Drainer state to `paused` and restart the Pump node.

4.15.13.18 How can I close a Pump or Drainer node?

Currently, you can only use the `offline-pump` or `offline-drainer` command in binlogctl to close a Pump or Drainer node.

4.15.13.19 When can I use the `update-pump` command in binlogctl to set the Pump state to `offline`?

You can use the `update-pump` command to set the Pump state to `offline` in the following situations:

- When a Pump process is exited abnormally and the service cannot be recovered, the replication task is interrupted. If you want to recover the replication and accept some losses of binlog data, use the `update-pump` command to set the Pump state to `offline` ↪ . Then, the Drainer node stops pulling binlog from the Pump node and continues replicating data.
- Some stale Pump nodes are left over from historical tasks. Their processes have been exited and their services are no longer needed. Then, use the `update-pump` command to set their state to `offline`.

For other situations, use the `offline-pump` command to close the Pump service, which is the regular process.

> **Warning:**
>
> Do not use the `update-pump` command unless you can tolerate binlog data loss and data inconsistency between upstream and downstream, or you no longer need the binlog data stored in the Pump node.

4.15.13.20 Can I use the `update-pump` command in binlogctl to set the Pump state to `offline` if I want to close a Pump node that is exited and set to `paused`?

When a Pump process is exited and the node is in the `paused` state, not all the binlog data in the node is consumed in its downstream Drainer node. Therefore, doing so might risk data inconsistency between upstream and downstream. In this situation, restart the Pump and use the `offline-pump` command to close the Pump node.

4.15.13.21 When can I use the `update-drainer` command in binlogctl to set the Drainer state to `offline`?

Some stale Drainer nodes are left over from historical tasks. Their processes have been exited and their services are no longer needed. Then, use the `update-drainer` command to set their state to `offline`.

4.15.13.22 Can I use SQL operations such as `change pump` and `change drainer` to pause or close the Pump or Drainer service?

No. For more details on these SQL operations, refer to Use SQL statements to manage Pump or Drainer.

These SQL operations directly modifies the state information saved in PD and are functionally equivalent to the `update-pump` and `update-drainer` commands in binlogctl. To pause or close the Pump or Drainer service, use the binlogctl tool.

4.15.13.23 What can I do when some DDL statements supported by the upstream database cause error when executed in the downstream database?

To solve the problem, follow these steps:

1. Check `drainer.log`. Search `exec failed` for the last failed DDL operation before the Drainer process is exited.
2. Change the DDL version to the one compatible to the downstream. Perform this step manually in the downstream database.
3. Check `drainer.log`. Search for the failed DDL operation and find the `commit-ts` of this operation. For example:

```
[2020/05/21 09:51:58.019 +08:00] [INFO] [syncer.go:398] ["add ddl item
    ↪ to syncer, you can add this commit ts to `ignore-txn-commit-ts`
    ↪ to skip this ddl if needed"] [sql="ALTER TABLE `test` ADD INDEX
    ↪ (`index1`)"] ["commit ts"=416815754209656834]。
```

4. Modify the `drainer.toml` configuration file. Add the `commit-ts` in the `ignore-txn-commit-ts` item and restart the Drainer node.

4.15.13.24 TiDB fails to write to binlog and gets stuck, and `listener stopped, waiting for manual stop` appears in the log

In TiDB v3.0.12 and earlier versions, the binlog write failure causes TiDB to report the fatal error. TiDB does not automatically exit but only stops the service, which seems like getting stuck. You can see the `listener stopped, waiting for manual stop` error in the log.

You need to determine the specific causes of the binlog write failure. If the failure occurs because binlog is slowly written into the downstream, you can consider scaling out Pump or increasing the timeout time for writing binlog.

Since v3.0.13, the error-reporting logic is optimized. The binlog write failure causes transaction execution to fail and TiDB Binlog will return an error but will not get TiDB stuck.

4.15.13.25 TiDB writes duplicate binlogs to Pump

This issue does not affect the downstream and replication logic.

When the binlog write fails or becomes timeout, TiDB retries writing binlog to the next available Pump node until the write succeeds. Therefore, if the binlog write to a Pump node

is slow and causes TiDB timeout (default 15s), then TiDB determines that the write fails and tries to write to the next Pump node. If binlog is actually successfully written to the timeout-causing Pump node, the same binlog is written to multiple Pump nodes. When Drainer processes the binlog, it automatically de-duplicates binlogs with the same TSO, so this duplicate write does not affect the downstream and replication logic.

4.15.13.26 Reparo is interrupted during the full and incremental restore process. Can I use the last TSO in the log to resume replication?

Yes. Reparo does not automatically enable the safe-mode when you start it. You need to perform the following steps manually:

1. After Reparo is interrupted, record the last TSO in the log as `checkpoint-tso`.
2. Modify the Reparo configuration file, set the configuration item `start-tso` to `checkpoint-tso + 1`, set `stop-tso` to `checkpoint-tso + 80,000,000,000` (approximately five minutes after the `checkpoint-tso`), and set `safe-mode` to `true`. Start Reparo, and Reparo replicates data to `stop-tso` and then stops automatically.
3. After Reparo stops automatically, set `start-tso` to `checkpoint tso + 80,000,000,001` ↪ , set `stop-tso` to `0`, and set `safe-mode` to `false`. Start Reparo to resume replication.

4.16 Tools

4.16.1 TiDB Tools Use Cases

This document introduces the common use cases of TiDB tools and how to choose the right tool for your scenario.

4.16.1.1 Deploy and operate TiDB in Kubernetes

If you need to deploy and operate TiDB in Kubernetes, you can deploy a Kubernetes cluster, and then deploy TiDB Operator. After that, you can use TiDB Operator to deploy and operate a TiDB cluster.

4.16.1.2 Import data from CSV to TiDB

If you need to import the compatible CSV files exported by other tools to TiDB, use TiDB Lightning.

4.16.1.3 Import full data from MySQL/Aurora

If you need to import full data from MySQL/Aurora, use Dumpling first to export data as SQL dump files, and then use TiDB Lightning to import data into the TiDB cluster.

4.16.1.4 Migrate data from MySQL/Aurora

If you need to migrate both full data and incremental data from MySQL/Aurora, use TiDB Data Migration (DM) to perform the full and incremental data migration.

If the full data volume is large (at the TB level), you can first use Dumpling and TiDB Lightning to perform the full data migration, and then use DM to perform the incremental data migration.

4.16.1.5 Back up and restore TiDB cluster

If you need to back up a TiDB cluster, use Dumpling.

If you need to restore data to a TiDB cluster, use TiDB Lightning.

4.16.1.6 Migrate data from TiDB

If you need to migrate data from a TiDB cluster to MySQL or to another TiDB cluster, use Dumpling to export full data from TiDB as SQL dump files, and then use TiDB Lightning to import data to MySQL or another TiDB cluster.

If you also need to migrate incremental data, use TiDB Binlog.

4.16.1.7 TiDB incremental data subscription

If you need to subscribe to TiDB's incremental changes, use TiDB Binlog.

4.16.2 Download Tools

This document collects the available downloads for most officially maintained versions of TiDB tools.

4.16.2.1 TiDB Operator

TiDB Operator runs in Kubernetes. After deploying the Kubernetes cluster, you can choose to deploy TiDB Operator either online or offline. For more information, see Deploying TiDB Operator in Kubernetes.

4.16.2.2 TiDB Binlog

If you want to download the latest version of TiDB Binlog, directly download the TiDB package, because TiDB Binlog is included in the TiDB package.

In addition, the Kafka version of TiDB Binlog is also provided.

Package name	OS	Architecture	SHA256 checksum
`https ↪ :// ↪ download ↪ . ↪ pingcap ↪ . ↪ org ↪ / ↪ tidb ↪ -{ ↪ version ↪ }- ↪ linux ↪ - ↪ amd64 ↪ . ↪ tar ↪ . ↪ gz ↪` (TiDB Binlog)	Linux	amd64	`https ↪ :// ↪ download ↪ . ↪ pingcap ↪ . ↪ org ↪ / ↪ tidb ↪ -{ ↪ version ↪ }- ↪ linux ↪ - ↪ amd64 ↪ . ↪ sha256 ↪`

Package name	OS	Architecture	SHA256 checksum
`https://download.pingcap.org/tidb-{version}-linux-amd64.tar.gz` (the Kafka version of TiDB Binlog)	Linux	amd64	`https://download.pingcap.org/tidb-{version}-linux-amd64.sha256`

Note:

`{version}` in the above download link indicates the version number of TiDB. For example, the download link for `v3.0.5` is `https://download.pingcap.org/tidb-v3.0.5-linux-amd64.tar.gz`.

4.16.2.3 TiDB Lightning

Download TiDB Lightning by using the download link in the following table:

Package name	OS	Architecture	SHA256 checksum
`https://download.pingcap.org/tidb-toolkit-{version}-linux-amd64.tar.gz`	Linux	amd64	`https://download.pingcap.org/tidb-toolkit-{version}-linux-amd64.sha256`

Note:

`{version}` in the above download link indicates the version number of TiDB Lightning. For example, the download link for `v3.0.5` is `https://download.pingcap.org/tidb-toolkit-v3.0.5-linux-amd64.tar.gz`.

4.16.2.4 TiDB DM (Data Migration)

Download DM by using the download link in the following table:

Package name	OS	Architecture	SHA256 checksum
`https://download.pingcap.org/dm-{version}-linux-amd64.tar.gz`	Linux	amd64	`https://download.pingcap.org/dm-{version}-linux-amd64.tar.gz.sha256`

Note:

`{version}` in the above download link indicates the version number of DM. For example, the download link for `v1.0.1` is `https://download.pingcap.org/dm-v1.0.1-linux-amd64.tar.gz`. You can check the published DM versions in the DM Release page.

4.16.2.5 Syncer, Loader, and Mydumper

If you want to download the latest version of Syncer, Loader, or Mydumper, directly download the tidb-enterprise-tools package, because all these tools are included in this package.

Package name	OS	Architecture	SHA256 checksum
tidb-enterprise-tools-nightly-linux-amd64.tar.gz	Linux	amd64	tidb-enterprise-tools-nightly-linux-amd64.sha256

This enterprise tools package includes all the following tools:

- Syncer
- Loader
- Mydumper
- ddl_checker
- sync-diff-inspector

4.16.3 TiDB Operator

TiDB Operator is an automatic operation system for TiDB clusters in Kubernetes. It provides full life-cycle management for TiDB including deployment, upgrades, scaling, backup, fail-over, and configuration changes. With TiDB Operator, TiDB can run seamlessly in the Kubernetes clusters deployed on a public or private cloud.

Currently, the TiDB Operator documentation (also named as TiDB in Kubernetes documentation) is independent of the TiDB documentation. To access the documentation, click the following link:

- TiDB in Kubernetes documentation

4.16.4 Table Filter

The TiDB ecosystem tools operate on all the databases by default, but oftentimes only a subset is needed. For example, you only want to work with the schemas in the form of `foo*` and `bar*` and nothing else.

Several TiDB ecosystem tools share a common filter syntax to define subsets. This document describes how to use the table filter feature.

4.16.4.1 Usage

4.16.4.1.1 CLI

Table filters can be applied to the tools using multiple `-f` or `--filter` command line parameters. Each filter is in the form of `db.table`, where each part can be a wildcard (further explained in the next section). The following lists the example usage in each tool.

- Dumpling:

```
./dumpling -f 'foo*.*' -f 'bar*.*' -P 3306 -o /tmp/data/
#          ^~~~~~~~~~~~~~~~~~~~~~
```

- Lightning:

```
./tidb-lightning -f 'foo*.*' -f 'bar*.*' -d /tmp/data/ --backend tidb
#                ^~~~~~~~~~~~~~~~~~~~~~
```

4.16.4.1.2 TOML configuration files

Table filters in TOML files are specified as array of strings. The following lists the example usage in each tool.

- Lightning:

```
[mydumper]
filter = ['foo*.*', 'bar*.*']
```

4.16.4.2 Syntax

4.16.4.2.1 Plain table names

Each table filter rule consists of a "schema pattern" and a "table pattern", separated by a dot (`.`). Tables whose fully-qualified name matches the rules are accepted.

```
db1.tbl1
db2.tbl2
db3.tbl3
```

A plain name must only consist of valid identifier characters, such as:

- digits (`0` to `9`)
- letters (`a` to `z`, `A` to `Z`)
- `$`
- `_`
- non ASCII characters (U+0080 to U+10FFFF)

All other ASCII characters are reserved. Some punctuations have special meanings, as described in the next section.

4.16.4.2.2 Wildcards

Each part of the name can be a wildcard symbol described in fnmatch(3):

- `*` — matches zero or more characters
- `?` — matches one character
- `[a-z]` — matches one character between "a" and "z" inclusively
- `[!a-z]` — matches one character except "a" to "z".

```
db[0-9].tbl[0-9a-f][0-9a-f]
data.*
*.backup_*
```

"Character" here means a Unicode code point, such as:

- U+00E9 (é) is 1 character.
- U+0065 U+0301 (é) are 2 characters.
- U+1F926 U+1F3FF U+200D U+2640 U+FE0F ( ) are 5 characters.

4.16.4.2.3 File import

To import a file as the filter rule, include an `@` at the beginning of the rule to specify the file name. The table filter parser treats each line of the imported file as additional filter rules.

For example, if a file `config/filter.txt` has the following content:

```
employees.*
*.WorkOrder
```

the following two invocations are equivalent:

```
./dumpling -f '@config/filter.txt'
./dumpling -f 'employees.*' -f '*.WorkOrder'
```

A filter file cannot further import another file.

4.16.4.2.4 Comments and blank lines

Inside a filter file, leading and trailing white-spaces of every line are trimmed. Furthermore, blank lines (empty strings) are ignored.

A leading `#` marks a comment and is ignored. `#` not at start of line is considered syntax error.

```
### this line is a comment
db.table  # but this part is not comment and may cause error
```

4.16.4.2.5 Exclusion

An `!` at the beginning of the rule means the pattern after it is used to exclude tables from being processed. This effectively turns the filter into a block list.

```
*.*
#^ note: must add the *.* to include all tables first
!*.Password
!employees.salaries
```

4.16.4.2.6 Escape character

To turn a special character into an identifier character, precede it with a backslash `\`.

```
db\.with\.dots.*
```

For simplicity and future compatibility, the following sequences are prohibited:

- `\` at the end of the line after trimming whitespaces (use `[ ]` to match a literal whitespace at the end).
- `\` followed by any ASCII alphanumeric character (`[0-9a-zA-Z]`). In particular, C-like escape sequences like `\0`, `\r`, `\n` and `\t` currently are meaningless.

4.16.4.2.7 Quoted identifier

Besides `\`, special characters can also be suppressed by quoting using `"` or `` ` ``.

```
"db.with.dots"."tbl\1"
`db.with.dots`.`tbl\2`
```

The quotation mark can be included within an identifier by doubling itself.

```
"foo""bar".`foo``bar`
### equivalent to:
foo\"bar.foo\`bar
```

Quoted identifiers cannot span multiple lines.

It is invalid to partially quote an identifier:

```
"this is "invalid*.*
```

4.16.4.2.8 Regular expression

In case very complex rules are needed, each pattern can be written as a regular expression delimited with `/`:

```
/^db\d{2,}$/./^tbl\d{2,}$/
```

These regular expressions use the Go dialect. The pattern is matched if the identifier contains a substring matching the regular expression. For instance, `/b/` matches `db01`.

> **Note:**
>
> Every `/` in the regular expression must be escaped as `\/`, including inside `[…]`. You cannot place an unescaped `/` between `\Q…\E`.

4.16.4.3 Multiple rules

When a table name matches none of the rules in the filter list, the default behavior is to ignore such unmatched tables.

To build a block list, an explicit `*.*` must be used as the first rule, otherwise all tables will be excluded.

```
### every table will be filtered out
./dumpling -f '!*.Password'

### only the "Password" table is filtered out, the rest are included.
./dumpling -f '*.*' -f '!*.Password'
```

In a filter list, if a table name matches multiple patterns, the last match decides the outcome. For instance:

```
### rule 1
employees.*
### rule 2
!*.dep*
### rule 3
*.departments
```

The filtered outcome is as follows:

Table name	Rule 1	Rule 2	Rule 3	Outcome
irrelevant.table				Default (reject)
employees.employees				Rule 1 (accept)
employees.dept_emp				Rule 2 (reject)
employees.departments				Rule 3 (accept)
else.departments				Rule 3 (accept)

Note:

In TiDB tools, the system schemas are always excluded regardless of the table filter settings. The system schemas are:

- `INFORMATION_SCHEMA`
- `PERFORMANCE_SCHEMA`
- `METRICS_SCHEMA`
- `INSPECTION_SCHEMA`
- `mysql`
- `sys`

4.16.5 Mydumper Instructions

4.16.5.1 What is Mydumper?

Mydumper is a fork project optimized for TiDB. You can use this tool for logical backups of **MySQL** or **TiDB**.

It can be downloaded as part of the Enterprise Tools package.

4.16.5.1.1 What enhancements does it contain over regular Mydumper?

- To ensure backup consistency for TiDB, this optimized Mydumper tool sets the value of tidb_snapshot to specify the point in time when the data is backed up instead of using `FLUSH TABLES WITH READ LOCK`.
- This tool uses the hidden `_tidb_rowid` column of TiDB to optimize the performance of concurrently exporting data from a single table.

4.16.5.2 Usage

4.16.5.2.1 New parameter description

`-z` or `--tidb-snapshot`: sets the `tidb_snapshot` to be used for the backup. The default value is the current TSO (the `Position` field output from `SHOW MASTER STATUS`). Set this parameter to the TSO or a valid `datetime` such as `-z "2016-10-08 16:45:26"`.

4.16.5.2.2 Required privileges

- SELECT
- RELOAD

- LOCK TABLES
- REPLICATION CLIENT

4.16.5.2.3 Usage example

Execute the following command to back up data from TiDB. You can add command line parameters to the command as needed:

```
./bin/mydumper -h 127.0.0.1 -u root -P 4000
```

4.16.5.3 Dump table data concurrently

This section introduces the working principle and parameters of Mydumper. This section also gives an example of Mydumper command, and explains the performance evaluation and the TiDB versions that support the `_tidb_rowid` index.

4.16.5.3.1 Working principle

Mydumper first calculates `min(_tidb_rowid)` and `max(_tidb_rowid)`, and segments the table into chunks according to the value specified by `-r`. Then, Mydumper assigns these chunks to different threads and dumps these chunks concurrently.

4.16.5.3.2 Parameters

- `-t` or `--threads`: specifies the number of concurrent threads (`4` by default).
- `-r` or `--rows`: specifies the maximum number of rows in a chunk. If this parameter is specified, Mydumper ignores the value of `--chunk-filesize`.

4.16.5.3.3 Example

The following is a complete Mydumper command:

```
./bin/mydumper -h 127.0.0.1 -u root -P 4000 -r 10000 -t 4
```

4.16.5.3.4 Performance evaluation

Do a performance evaluation before you perform the dump operation. Because the concurrent scanning brings pressure on the TiDB and TiKV clusters, you need to evaluate and test the impact that the dump operation might have on the database clusters and applications.

4.16.5.3.5 TiDB versions that support the `_tidb_rowid` index

Because concurrent table data dump uses the implicit `_tidb_rowid` row of TiDB, TiDB versions that support the `_tidb_rowid` index can fully take advantage of the concurrent dump.

The following TiDB versions supports the `_tidb_rowid` index:

- v2.1.3 and later v2.1 versions
- v3.0 (by default)

4.16.5.4 FAQ

4.16.5.4.1 How to resolve the error that occurs when the `--tidb-snapshot` option is used to export data?

In this situation, you need to add a `--skip-tz-utc` option. Otherwise, Mydumper will pre-configure the UTC time zone and convert the time zone when `tidb-snapshot` is configured, which causes this error.

4.16.5.4.2 How to determine if the Mydumper I am using is the PingCAP optimized version?

Execute the following command:

```
./bin/mydumper -V
```

If the output contains `git_hash` (`d3e6fec8b069daee772d0dbaa47579f67a5947e7` in the following example), you are using the PingCAP optimized version of Mydumper:

```
mydumper 0.9.5 (d3e6fec8b069daee772d0dbaa47579f67a5947e7), built against
    ↪ MySQL 5.7.24
```

4.16.5.4.3 How to resolve the "invalid mydumper files for there are no `-schema-create.sql` files found" error when using Loader to restore the data backed up by Mydumper?

Check whether the `-T` or `--tables-list` option is used when using Mydumper to back up data. If these options are used, Mydumper does not generate a file that includes a `CREATE DATABASE` SQL statement.

Solution: Create the `{schema-name}-schema-create.sql` file in the directory for data backup of Mydumper. Write "CREATE DATABASE `{schema-name}`" to the file, and then run Loader.

4.16.5.4.4 Why is the TIMESTAMP type of data exported using Mydumper inconsistent with that in the database?

Check whether the time zone of the server that is running Mydumper is consistent with that of the database. Mydumper converts the TIMESTAMP type of data according to the time zone of its server. You can add the `--skip-tz-utc` option to disable the conversion of dates and times.

4.16.5.4.5 How to configure the `-F,--chunk-filesize` option of Mydumper?

Mydumper splits the data of each table into multiple chunks according to the value of this option during backup. Each chunk is saved in a file with a size of about `chunk-filesize` ↪ . In this way, data is split into multiple files and you can use the parallel processing of Loader/TiDB lightning to improve the import speed. If you later use **Loader** to restore the backup files, it is recommended to set the value of this option to `64` (in MB); If you use **TiDB Lightning** to restore files, `256` (in MB) is recommended.

4.16.5.4.6 How to configure the `-s --statement-size` option of Mydumper?

Mydumper uses this option to control the size of `Insert Statement` which defaults to `10000000` (about 1 MB). Use this option to avoid the following errors when restoring data:

```
packet for query is too large. Try adjusting the 'max_allowed_packet'
    ↪ variable
```

The default value meets the requirements in most cases, but **if it is a wide table, the size of a single row of data might exceed the limit of `statement-size`, and Mydumper reports the following warning**:

```
Row bigger than statement_size for xxx
```

If you restore the data in this situation, Mydumper still reports the `packet for query` ↪ `is too large` error. To solve this problem, modify the following two configurations (take `128 MB` as an example):

- Execute `set @@global.max_allowed_packet=134217728` (`134217728` = `128 MB`) in TiDB server.
- Add the `max-allowed-packet=128M` line to the DB configuration of Loader or DM task's configuration file according to your situation. Then, restart the process or task.

4.16.5.4.7 How to set the `-l, --long-query-guard` option of Mydumper?

Set the value of this option to the estimated time required for a backup. If Mydumper runs longer than this value, it reports an error and exits. It is recommended to set the value to `7200` (in seconds) for the first time of your backup and then modify it according to your actual backup time.

4.16.5.4.8 How to set the `--tidb-force-priority` option of Mydumper?

This option can only be set when backing up TiDB's data. It can be set to `LOW_PRIORITY` , `DELAYED`, or `HIGH_PRIORITY`. If you do not want data backup to affect online services, it is recommended to set this option to `LOW_PRIORITY`; if the backup has a higher priority, `HIGH_PRIORITY` is recommended.

4.16.5.4.9 How to resolve the "GC life time is short than transaction duration" error when using Mydumper to back up TiDB's data?

Mydumper uses the `tidb_snapshot` system variable to ensure data consistency when backing up TiDB's data. This error is reported if the historical data of a snapshot is cleared by TiDB's Garbage Collection (GC) during backup. To solve this problem, perform the following steps:

1. Before using Mydumper to back up data, use MySQL client to check the value of `tikv_gc_life_time` in the TiDB cluster and set it to an appropriate value:

```
SELECT * FROM mysql.tidb WHERE VARIABLE_NAME = 'tikv_gc_life_time';
```

```
+-----------------------+------------------------------------------------------------
    ↪
| VARIABLE_NAME       | VARIABLE_VALUE
    ↪
    ↪ |
+-----------------------+------------------------------------------------------------
    ↪
| tikv_gc_life_time   | 10m0s
    ↪
    ↪ |
+-----------------------+------------------------------------------------------------
    ↪
1 rows in set (0.02 sec)
```

```
update mysql.tidb set VARIABLE_VALUE = '720h' where VARIABLE_NAME = '
    ↪ tikv_gc_life_time';
```

2. Set the value of `tikv_gc_life_time` to the initial one after the backup is complete:

```
update mysql.tidb set VARIABLE_VALUE = '10m0s' where VARIABLE_NAME = '
    ↪ tikv_gc_life_time';
```

4.16.5.4.10 Do I need to configure the `--tidb-rowid` option of Mydumper?

If this option is set to true, the exported data contains the data of TiDB's hidden columns. Using hidden columns when restoring data to TiDB might cause data inconsistency. Currently, it is not recommended to use this option.

4.16.5.4.11 How to resolve the "Segmentation Fault" error?

This bug has been fixed. If the error persists, you can upgrade to the latest version of Mydumper.

4.16.5.4.12 How to resolve the "Error dumping table ({schema}.{table}) data: line ..... (total length ...)" error?

This error occurs when Mydumper parses SQL statements. In this situation, use the latest version of Mydumper. If this error persists, you can file an issue to mydumper/issues.

4.16.5.4.13 How to resolve the "Failed to set tidb_snapshot: parsing time "20190901-10:15:00 +0800" as "20190901-10:15:00 +0700 MST": cannot parse "" as "MST"" error?

Check whether the version of TiDB is lower than 2.1.11. If so, upgrade to TiDB 2.1.11 or later versions.

4.16.5.4.14 Do you plan to make these changes available to upstream Mydumper?

Yes, we intend to make our changes available to upstream Mydumper. See PR #155.

4.16.6 Syncer User Guide

> **Warning:**
>
> Syncer is no longer maintained. Its features are completely superseded by TiDB Data Migration. It is strongly recommended that you use TiDB Data Migration instead.

4.16.6.1 About Syncer

Syncer is a tool used to import data incrementally. It is a part of the TiDB enterprise toolset.

It can be downloaded as part of the Enterprise Tools package.

4.16.6.2 Syncer architecture

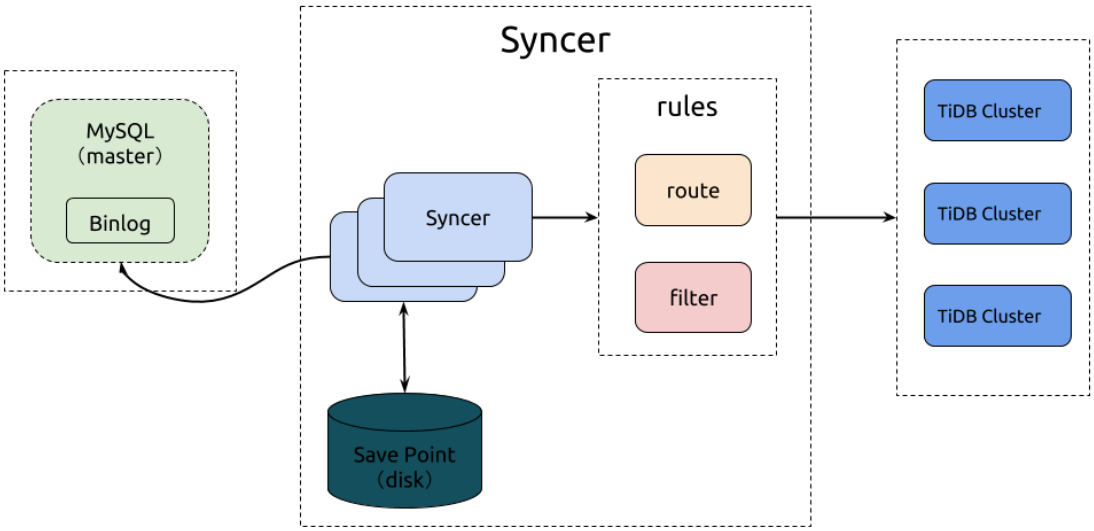

Figure 332: syncer sharding

4.16.6.3 Where to deploy Syncer

You can deploy Syncer to any of the machines that can connect to MySQL or the TiDB cluster. But it is recommended to deploy Syncer to the TiDB cluster.

4.16.6.4 Use Syncer to import data incrementally

Before importing data, read Check before importing data using Syncer.

4.16.6.4.1 1. Set the position to replicate

Edit the meta file of Syncer, assuming the meta file is `syncer.meta`:

```
### cat syncer.meta
binlog-name = "mysql-bin.000003"
binlog-pos = 930143241
binlog-gtid = "2bfabd22-fff7-11e6-97f7-f02fa73bcb01:1-23,61ccbb5d-c82d-11e6-
    ↪ ac2e-487b6bd31bf7:1-4"
```

Note:

- The `syncer.meta` file only needs to be configured when it is first used. The position is automatically updated when the new subsequent binlog is replicated.
- If you use the binlog position to replicate, you only need to configure `binlog-name` and `binlog-pos`; if you use `binlog-gtid` to replacate, you need to configure `binlog-gtid` and set `--enable-gtid` when starting Syncer.

4.16.6.4.2 2. Start Syncer

Description of Syncer command line options:

```
Usage of syncer:
  -L string
        log level: debug, info, warn, error, fatal (default "info")
  -V    to print Syncer version info (default false)
  -b int
        the size of batch transactions (default 100)
  -c int
        the number of batch threads that Syncer processes (default 16)
  -config string
        to specify the corresponding configuration file when starting Syncer;
           ↪  for example, `--config config.toml`
  -enable-ansi-quotes
        to enable ANSI_QUOTES sql_mode
  -enable-gtid
        to start Syncer using the mode; default false; before enabling this
           ↪ option, you need to enable GTID in the upstream MySQL
  -flavor string
        use flavor for different MySQL source versions; support "mysql", "
           ↪ mariadb" now; if you replicate data from MariaDB, set it to "
           ↪ mariadb" (default "mysql")
  -log-file string
        to specify the log file directory, such as `--log-file ./syncer.log`
  -log-rotate string
        to specify the log file rotating cycle, hour/day (default "day")
  -max-retry int
        to specify the maximum times an SQL statement should be retried. One
           ↪ common cause of statement retries is network interruption (
           ↪ default 100)
```

```
  -meta string
        to specify the meta file of the upstream of Syncer (in the same
           ↪ directory with the configuration file, "syncer.meta" by
           ↪ default)
  -persistent-dir string
        to specify the persistent file (historical reason: it is not a
           ↪ directory) of Syncer history table schemas; if you set it to a
           ↪  non-empty string, the history table schema is chosen
           ↪ according to the column length when you construct DML
           ↪ statements
  -safe-mode
        to specify and enable the safe mode to make Syncer reentrant
  -server-id int
        to specify MySQL replica sever-id (default 101)
  -status-addr string
        to specify Syncer metrics (default :8271), such as `--status-addr
           ↪ 127:0.0.1:8271`
  -timezone string
        the time zone used by the target database; it is required to use the
           ↪ IANA time zone identifier such as `Asia/Shanghai`
```

The `config.toml` configuration file of Syncer:

```
log-level = "info"
log-file = "syncer.log"
log-rotate = "day"

server-id = 101

### The file path for meta:
meta = "./syncer.meta"
worker-count = 16
batch = 100
flavor = "mysql"

### It can be used by Prometheus to pull Syncer metrics, and is also the
   ↪ pprof address of Syncer.
status-addr = ":8271"

### If you set its value to true, Syncer stops and exits when it encounters
   ↪ the DDL operation.
stop-on-ddl = false

### The maximum number of times an SQL statement should be retried. One
   ↪ common cause of statement retries is network interruption.
```

```
max-retry = 100

### Specify the time zone used by the target database; all timestamp fields
    ↪ in binlog are converted according to the time zone; the local time
    ↪ zone of Syncer is used by default.
### timezone = "Asia/Shanghai"

### Skip the DDL statement; the format is **prefix exact match**, for
    ↪ example, you need to fill at least `DROP TABLE` in to skip `DROP
    ↪ TABLE ABC`.
### skip-ddls = ["ALTER USER", "CREATE USER"]

### After Syncer uses `route-rules` to map the upstream schema and table
    ↪ into `target-schema` and `target-table`,
### Syncer matches the mapped `target-schema` and `target-table` with do/
    ↪ ignore rules,
### and the matching sequence is: replicate-do-db --> replicate-do-table -->
    ↪  replicate-ignore-db --> replicate-ignore-table.
### Specify the database name to be replicated. Support regular expressions.
    ↪  Start with '~' to use regular expressions.
### replicate-do-db = ["~^b.*","s1"]

### Specify the database you want to ignore in replication. Support regular
    ↪ expressions. Start with '~' to use regular expressions.
### replicate-ignore-db = ["~^b.*","s1"]

### skip-dmls skips the DML binlog events. The type value can be 'insert', '
    ↪ update' and 'delete'.
### The 'delete' statements that skip-dmls skips in the foo.bar table:
### [[skip-dmls]]
### db-name = "foo"
### tbl-name = "bar"
### type = "delete"
### # The 'delete' statements that skip-dmls skips in all tables:
### [[skip-dmls]]
### type = "delete"
### # The 'delete' statements that skip-dmls skips in all foo.* tables:
### [[skip-dmls]]
### db-name = "foo"
### type = "delete"

### Specify the db.table to be replicated.
### db-name and tbl-name do not support the `db-name ="dbname, dbname2"`
    ↪ format.
### [[replicate-do-table]]
```

```
### db-name ="dbname"
### tbl-name = "table-name"

### [[replicate-do-table]]
### db-name ="dbname1"
### tbl-name = "table-name1"

### Specify the db.table to be replicated. Support regular expressions.
    ↪ Start with '~' to use regular expressions.
### [[replicate-do-table]]
### db-name ="test"
### tbl-name = "~^a.*"

### Specify the database table you want to ignore in replication.
### db-name and tbl-name do not support the `db-name ="dbname, dbname2"`
    ↪ format.
### [[replicate-ignore-table]]
### db-name = "your_db"
### tbl-name = "your_table"

### Specify the database table you want to ignore in replication. Support
    ↪ regular expressions. Start with '~' to use regular expressions.
### [[replicate-ignore-table]]
### db-name ="test"
### tbl-name = "~^a.*"

### The sharding replicating rules support wildcharacter.
### 1. The asterisk character ("*", also called "star") matches zero or more
    ↪  characters,
###    For example, "doc*" matches "doc" and "document" but not "dodo";
###    The asterisk character must be in the end of the wildcard word,
###    and there is only one asterisk in one wildcard word.
### 2. The question mark ("?") matches any single character.
### [[route-rules]]
### pattern-schema = "route_*"
### pattern-table = "abc_*"
### target-schema = "route"
### target-table = "abc"

### [[route-rules]]
### pattern-schema = "route_*"
### pattern-table = "xyz_*"
### target-schema = "route"
### target-table = "xyz"
```

```
[from]
host = "127.0.0.1"
user = "root"
password = ""
port = 3306

[to]
host = "127.0.0.1"
user = "root"
password = ""
port = 4000
```

Start Syncer:

```
./bin/syncer -config config.toml

2016/10/27 15:22:01 binlogsyncer.go:226: [info] begin to sync binlog from
    ↪ position (mysql-bin.000003, 1280)
2016/10/27 15:22:01 binlogsyncer.go:130: [info] register slave for master
    ↪ server 127.0.0.1:3306
2016/10/27 15:22:01 binlogsyncer.go:552: [info] rotate to (mysql-bin.000003,
    ↪  1280)
2016/10/27 15:22:01 syncer.go:549: [info] rotate binlog to (mysql-bin
    ↪ .000003, 1280)
```

4.16.6.4.3 3. Insert data into MySQL

```
INSERT INTO t1 VALUES (4, 4), (5, 5);
```

4.16.6.4.4 4. Log in to TiDB and view the data

```
mysql -h127.0.0.1 -P4000 -uroot -p
mysql> select * from t1;
+----+------+
| id | age |
+----+------+
|  1 |   1 |
|  2 |   2 |
|  3 |   3 |
|  4 |   4 |
|  5 |   5 |
+----+------+
```

Syncer outputs the current replicated data statistics every 30 seconds:

```
2017/06/08 01:18:51 syncer.go:934: [info] [syncer]total events = 15, total
    ↪ tps = 130, recent tps = 4,
master-binlog = (ON.000001, 11992), master-binlog-gtid=53ea0ed1-9bf8-11e6-8
    ↪ bea-64006a897c73:1-74,
syncer-binlog = (ON.000001, 2504), syncer-binlog-gtid = 53ea0ed1-9bf8-11e6-8
    ↪ bea-64006a897c73:1-17
2017/06/08 01:19:21 syncer.go:934: [info] [syncer]total events = 15, total
    ↪ tps = 191, recent tps = 2,
master-binlog = (ON.000001, 11992), master-binlog-gtid=53ea0ed1-9bf8-11e6-8
    ↪ bea-64006a897c73:1-74,
syncer-binlog = (ON.000001, 2504), syncer-binlog-gtid = 53ea0ed1-9bf8-11e6-8
    ↪ bea-64006a897c73:1-35
```

The update in MySQL is automatically replicated in TiDB.

4.16.6.5 Description of Syncer configuration

4.16.6.5.1 Specify the database to be replicated

This section describes the priority of parameters when you use Syncer to replicate the database.

- To use the route-rules, see Support for replicating data from sharded tables.
- Priority: replicate-do-db –> replicate-do-table –> replicate-ignore-db –> replicate-ignore-table

```
### Specify the ops database to be replicated.
### Specify to replicate the database starting with ti.
replicate-do-db = ["ops","~^ti.*"]

### The "china" database includes multiple tables such as guangzhou,
    ↪ shanghai and beijing. You only need to replicate the shanghai and
    ↪ beijing tables.
### Specify to replicate the shanghai table in the "china" database.
[[replicate-do-table]]
db-name ="china"
tbl-name = "shanghai"

### Specify to replicate the beijing table in the "china" database.
[[replicate-do-table]]
db-name ="china"
tbl-name = "beijing"
```

```
### The "ops" database includes multiple tables such as ops_user, ops_admin,
   ↪  weekly. You only need to replicate the ops_user table.
### Because replicate-do-db has a higher priority than replicate-do-table,
   ↪ it is invalid if you only set to replicate the ops_user table. In
   ↪ fact, the whole "ops" database is replicated.
[[replicate-do-table]]
db-name ="ops"
tbl-name = "ops_user"

### The "history" database includes multiple tables such as 2017_01 2017_02
   ↪ ... 2017_12/2016_01 2016_02 ... 2016_12. You only need to replicate
   ↪ the tables of 2017.
[[replicate-do-table]]
db-name ="history"
tbl-name = "~^2017_.*"

### Ignore the "ops" and "fault" databases in replication
### Ignore the databases starting with "www" in replication
### Because replicate-do-db has a higher priority than replicate-ignore-db,
   ↪ it is invalid to ignore the "ops" database here in replication.
replicate-ignore-db = ["ops","fault","~^www"]

### The "fault" database includes multiple tables such as faults,
   ↪ user_feedback, ticket.
### Ignore the user_feedback table in replication.
### Because replicate-ignore-db has a higher priority than replicate-ignore-
   ↪ table, it is invalid to only ignore the user_feedback table in
   ↪ replication. In fact, the whole "fault" database is ignored in
   ↪ replication.
[[replicate-ignore-table]]
db-name = "fault"
tbl-name = "user_feedback"

### The "order" database includes multiple tables such as 2017_01 2017_02
   ↪ ... 2017_12/2016_01 2016_02 ... 2016_12. You need to ignore the
   ↪ tables of 2016.
[[replicate-ignore-table]]
db-name ="order"
tbl-name = "~^2016_.*"
```

4.16.6.5.2 Support for replicating data from sharded tables

You can use Syncer to import data from sharded tables into one table within one database according to the `route-rules`. But before replicating, you need to check:

- Whether the sharding rules can be represented using the `route-rules` syntax.
- Whether the sharded tables contain unique increasing primary keys, or whether conflicts exist in the unique indexes or the primary keys after the combination.

Currently, the support for DDL is still in progress.

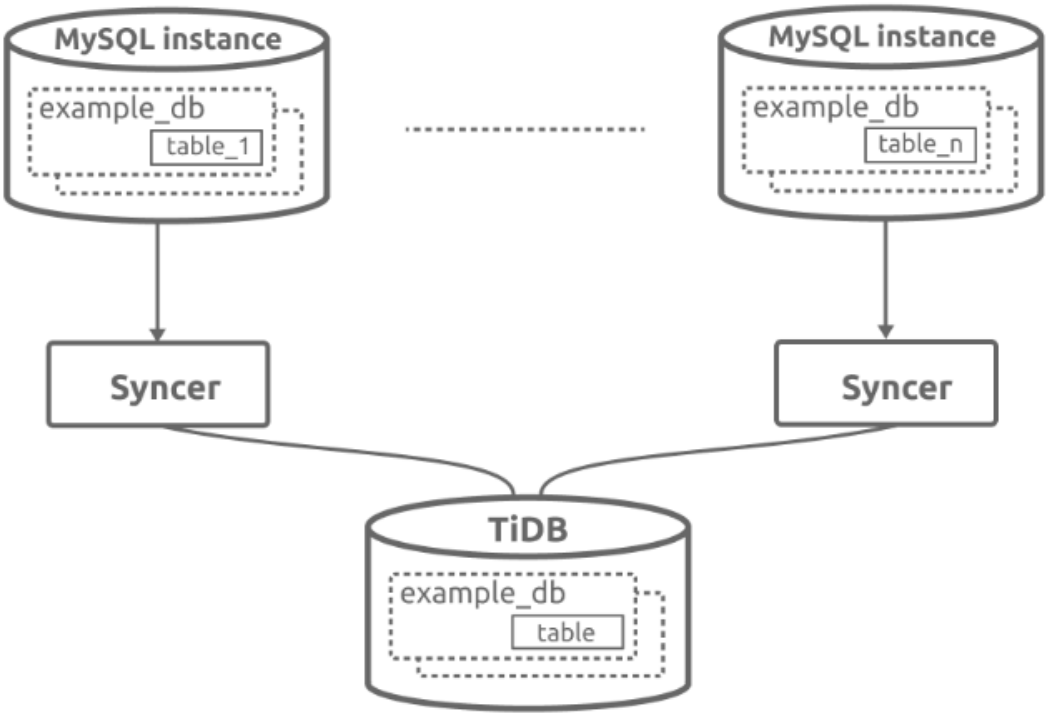

Figure 333: syncer sharding

Usage of replicating data from sharded tables

1. Start Syncer in all MySQL instances and configure the route-rules.
2. In scenarios using replicate-do-db & replicate-ignore-db and route-rules at the same time, you need to specify the target-schema & target-table content in route-rules.

```
### The scenarios are as follows:
### Database A includes multiple databases such as order_2016 and
    ↪ history_2016.
### Database B includes multiple databases such as order_2017 and
    ↪ history_2017.
```

```
### Specify to replicate order_2016 in database A; the data tables are 2016
    ↪ _01 2016_02 ... 2016_12
### Specify to replicate order_2017 in database B; the data tables are 2017
    ↪ _01 2017_02 ... 2017_12
### Use order_id as the primary key in the table, and the primary keys among
    ↪  data do not conflict.
### Ignore the history_2016 and history_2017 databases in replication
### The target database is "order" and the target data tables are order_2017
    ↪  and order_2016.

### When Syncer finds that the route-rules is enabled after Syncer gets the
    ↪ upstream data, it first combines databases and tables, and then
    ↪ determines do-db & do-table.
### You need to configure the database to be replicated, which is required
    ↪ when you determine the target-schema & target-table.
[[replicate-do-table]]
db-name ="order"
tbl-name = "order_2016"

[[replicate-do-table]]
db-name ="order"
tbl-name = "order_2017"

[[route-rules]]
pattern-schema = "order_2016"
pattern-table = "2016_??"
target-schema = "order"
target-table = "order_2016"

[[route-rules]]
pattern-schema = "order_2017"
pattern-table = "2017_??"
target-schema = "order"
target-table = "order_2017"
```

4.16.6.5.3 Check before replicating data using Syncer

Before replicating data using Syncer, check the following items:

1. Check the database version.

   Use the `select @@version;` command to check your database version. Currently, Syncer supports the following versions:

   - 5.5 < MySQL version < 8.0

- MariaDB version >= 10.1.2

  In earlier versions of MariaDB, the format of some binlog field types is inconsistent with that in MySQL.

> **Note:**
>
> If there is a source/replica replication structure between the upstream MySQL/MariaDB servers, then choose the following version.
>
> - 5.7.1 < MySQL version < 8.0
> - MariaDB version >= 10.1.3

2. Check the `server-id` of the source database.

   Check the `server-id` using the following command:

```
mysql> show global variables like 'server_id';
+---------------+-------
| Variable_name | Value |
+---------------+-------+
| server_id     | 1     |
+---------------+-------+
1 row in set (0.01 sec)
```

   - If the result is null or 0, Syncer cannot replicate data.
   - Syncer `server-id` must be different from the MySQL `server-id`, and must be unique in the MySQL cluster.

3. Check binlog related parameters.

   1. Check whether the binlog is enabled in MySQL using the following command:

```
mysql> show global variables like 'log_bin';
+--------------------+---------+
| Variable_name      | Value   |
+--------------------+---------+
| log_bin            | ON      |
+--------------------+---------+
1 row in set (0.00 sec)
```

      If the result is `log_bin = OFF`, you need to enable the binlog. See the document about enabling the binlog.

   2. The binlog format must be `ROW` and the binary log must be written with `FULL` row images. Check both of these variables:

```
mysql> select variable_name, variable_value from
    ↪ information_schema.global_variables where variable_name in (
    ↪ 'binlog_format','binlog_row_image');
+-----------------+----------------+
| variable_name   | variable_value |
+-----------------+----------------+
| BINLOG_FORMAT   | ROW            |
| BINLOG_ROW_IMAGE | FULL          |
+-----------------+----------------+
2 rows in set (0.001 sec)
```

- If one of the settings is not correct, you should change the configuration file on disk and restart the MySQL service.
- It's important to persist any configuration changes to disk, so that they're reflected if the MySQL service restarts.
- Because existing connections will keep old values of global variables, you should *not* use the `SET` statement to dynamically change these settings.

4. Check user privileges.

    1. Check the user privileges required by Mydumper for full data export.

        - To export the full data using Mydumper, the user must have the privileges of `select` and `reload`.
        - You can add the `--no-locks` option when the operation object is RDS, to avoid applying for the `reload` privilege.

    2. Check the upstream MySQL or MariaDB user privileges required by Syncer for incremental replication.

        The upstream MySQL user must have the following privileges at least:

        ```
        select, replication slave, replication client
        ```

    3. Check the downstream user privileges required by TiDB.

Privileges	Scope
`SELECT`	Tables
`INSERT`	Tables
`UPDATE`	Tables
`DELETE`	Tables
`CREATE`	Databases, tables
`DROP`	Databases, tables
`ALTER`	Tables
`INDEX`	Tables

Execute the following `GRANT` statement for the databases or tables that you need to replicate:

```
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER,INDEX ON db.
    ↪ table TO 'your_user'@'your_wildcard_of_host';
```

5. Check the SQL mode.

   Make sure that the upstream SQL mode is consistent with the downstream SQL mode. Otherwise, the data replication error might occur.

```
mysql> show variables like '%sql_mode%';
+--
    ↪ -------------+-----------------------------------------------------------
    ↪
| Variable_name | Value
    ↪                                                                      |
+--
    ↪ -------------+-----------------------------------------------------------
    ↪
| sql_mode      | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,
    ↪ NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+--
    ↪ -------------+-----------------------------------------------------------
    ↪
1 row in set (0.01 sec)
```

6. Check the Character Set.

   TiDB differs from MySQL in Character Set.

7. Check whether the table to be replicated has a primary key or a unique index.

   If the table does not have a primary key or a unique index, idempotent operations cannot be achieved. In this situation, a full table scan is performed every time an entry of data is updated in the downstream, which might slow down the replication task. Therefore, it is recommended that you add a primary key to every table to be replicated.

4.16.6.6 Syncer monitoring solution

The `syncer` monitoring solution contains the following components:

- Prometheus, an open source time series database, used to store the monitoring and performance metrics
- Grafana, an open source project for analyzing and visualizing metrics, used to display the performance metrics

- Alertmanager, combined with Grafana to implement the alerting mechanism

See the following diagram:

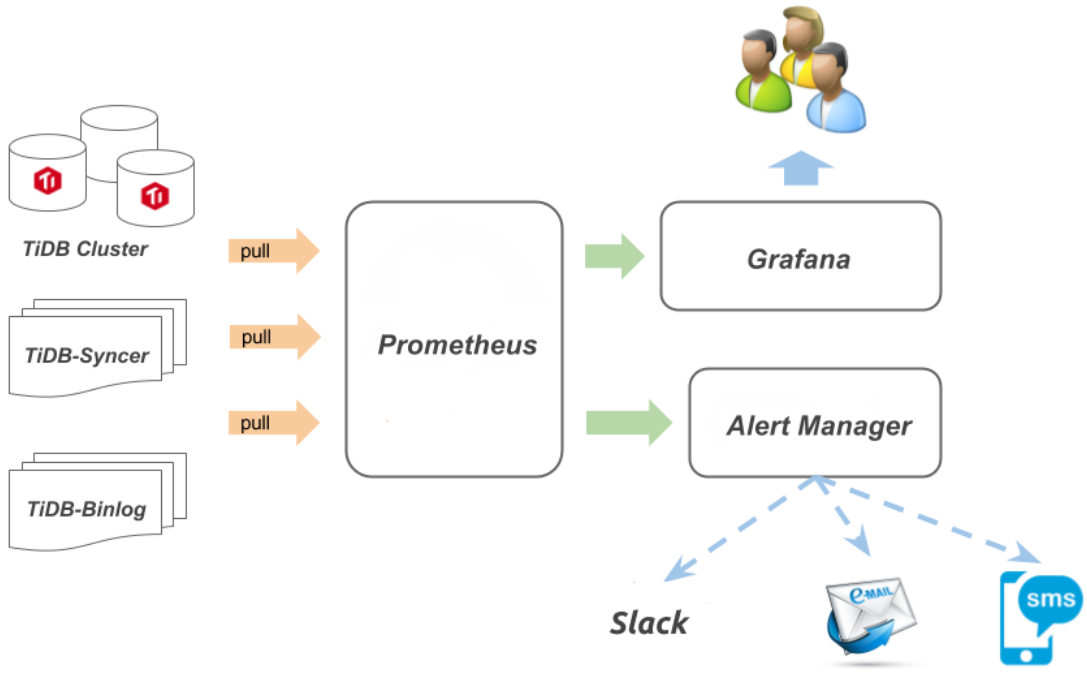

Figure 334: syncer_monitor_scheme

4.16.6.6.1 Configure Syncer monitor and alert

Syncer provides the metric interface, and requires Prometheus to actively obtain data. Take the following steps to configure Syncer monitor and alert:

1. To add the Syncer job information to Prometheus, flush the following content to the configuration file of Prometheus. The monitor is enabled when you restart Prometheus.

```
- job_name: 'syncer_ops' // name of the job, to distinguish the
    ↪ reported data
  static_configs:
    - targets: ['10.1.1.4:10086'] // Syncer monitoring address and
       ↪ port; to inform Prometheus of obtaining the monitoring
       ↪ data of Syncer
```

2. To configure Prometheus alert, flush the following content to the `alert.rule` configuration file. The alert is enabled when you restart Prometheus.

```
# syncer
ALERT syncer_status
  IF  syncer_binlog_file{node='master'} - ON(instance, job)
     ↪ syncer_binlog_file{node='syncer'} > 1
  FOR 1m
  LABELS {channels="alerts", env="test-cluster"}
  ANNOTATIONS {
  summary = "syncer status error",
  description="alert: syncer_binlog_file{node='master'} - ON(instance,
     ↪ job) syncer_binlog_file{node='syncer'} > 1 instance: {{ $labels
     ↪ .instance }} values: {{ $value }}",
  }
```

Configure Grafana

1. Log in to the Grafana Web interface.

   - The default address is: http://localhost:3000
   - The default account name: admin
   - The password for the default account: admin

2. Import the configuration file of Grafana dashboard.

   Click the Grafana Logo -> click Dashboards -> click Import -> choose and import the dashboard configuration file -> choose the corresponding data source.

4.16.6.6.2 Description of Grafana Syncer metrics

title: binlog events

- metrics: `rate(syncer_binlog_event_count[1m])`
- info: QPS of the binlog event that has been received by Syncer

title: binlog event transform

- metrics: `histogram_quantile(0.8, sum(rate(syncer_binlog_event_bucket[1m
  ↪]))by (le))`
- info: the cost of transforming the binlog event to SQL statements by Syncer

title: transaction latency

- metrics: `histogram_quantile(0.95, sum(rate(syncer_txn_cost_in_second_bucket [1m]))by (le))`
- info: the cost of executing a transaction on TiDB

title: transaction tps

- metrics: `rate(syncer_txn_cost_in_second_count[1m])`
- info: TPS of executing a transaction on TiDB

title: binlog file gap

- metrics: `syncer_binlog_file{node="master"} - ON(instance, job)syncer_binlog_file {node="syncer"}`
- info: the number of different binlog files between the upstream and the downstream in the process of replication; the normal value is 0, which indicates real-time replication; a larger value indicates a larger number of binlog files discrepancy

title: binlog skipped events

- metrics: `rate(syncer_binlog_skipped_events_total[1m])`
- info: the total number of SQL statements that Syncer skips when the upstream replicates binlog files with the downstream; you can configure the format of SQL statements skipped by Syncer using the `skip-ddls` and `skip-dmls` parameters in the `syncer.toml` file.

title: position binlog position

- metrics: `syncer_binlog_pos{node="syncer"}` and `syncer_binlog_pos{node="master"}`
- info: it works with `file number of binlog position`. `syncer_binlog_pos{node="master"}` indicates the position of latest binlog position fetched from MySQL, and `syncer_binlog_pos{node="syncer"}` indicates the position of the binlog position that Syncer has replicated.

title: file number of binlog position

- metrics: `syncer_binlog_file{node="syncer"}` and `syncer_binlog_file{node="master"}`
- info: it works with `position of binlog position`. `syncer_binlog_file{node="master"}` indicates the file number of the latest binlog position fetched from MySQL, and `syncer_binlog_file{node="syncer"}` indicates the file number of the binlog position that Syncer has replicated.

title: execution jobs

- metrics: `sum(rate(syncer_add_jobs_total[1m]))by (queueNo)`
- info: the count of jobs that have been added into the execution queue

title: pending jobs

- metrics: `sum(rate(syncer_add_jobs_total[1m])- rate(syncer_finished_jobs_total [1m]))by (queueNo)`
- info: the count of jobs that have been applied into TiDB

4.16.7 Loader Instructions

4.16.7.1 What is Loader?

Loader is a data import tool to load data to TiDB.

It can be downloaded as part of the tidb-enterprise-tools package.

4.16.7.2 Why did we develop Loader?

Since tools like mysqldump will take us days to migrate massive amounts of data, we used the Mydumper/myloader suite to multi-thread export and import data. During the process, we found that Mydumper works well. However, as myloader lacks functions of error retry and savepoint, it is inconvenient for us to use. Therefore, we developed loader, which reads the output data files of Mydumper and imports data to TiDB through the MySQL protocol.

4.16.7.3 What can Loader do?

- Multi-thread import data
- Support table level concurrent import and scattered hot spot write
- Support concurrent import of a single large table and scattered hot spot write
- Support Mydumper data format
- Support error retry
- Support savepoint
- Improve the speed of importing data through system variable

4.16.7.4 Usage

Note:

- Do not import the `mysql` system database from the MySQL instance to the downstream TiDB instance.
- If Mydumper uses the `-m` parameter, the data is exported without the table schema and the loader can not import the data.
- If you use the default `checkpoint-schema` parameter, after importing the data of a database, run `drop database tidb_loader` before you begin to import the next database.
- It is recommended to specify the `checkpoint-schema = "tidb_loader ↪ "` parameter when importing data.

4.16.7.4.1 Parameter description

```
-L string: the log level setting, which can be set as debug, info, warn,
   ↪ error, fatal (default: "info")

-P int: the port of TiDB (default: 4000)

-V boolean: prints version and exit

-c string: config file

-checkpoint-schema string: the database name of checkpoint. In the
   ↪ execution process, loader will constantly update this database.
   ↪ After recovering from an interruption, loader will get the process
   ↪ of the last run through this database. (default: "tidb_loader")

-d string: the storage directory of data that need to import (default:
   ↪ "./")

-h string: the host of TiDB (default: "127.0.0.1")

-p string: the account and password of TiDB

-status-addr string: It can be used by Prometheus to pull Loader metrics,
   ↪ and is also the pprof address of Loader (default: ":8272")

-t int: the number of thread,increase this as TiKV nodes increase (default
   ↪ : 16)
```

```
  -u string: the user name of TiDB (default: "root")
```

4.16.7.4.2 Configuration file

Apart from command line parameters, you can also use configuration files. The format is shown as below:

```
### Loader log level, which can be set as "debug", "info", "warn", "error"
    ↪ and "fatal" (default: "info")
log-level = "info"

### Loader log file
log-file = "loader.log"

### Directory of the dump to import (default: "./")
dir = "./"

### It can be used by Prometheus to pull Loader metrics, and is also the
    ↪ pprof address of Loader (default: ":8272").
status-addr = ":8272"

### The checkpoint data is saved to TiDB, and the schema name is defined
    ↪ here.
checkpoint-schema = "tidb_loader"

### Number of threads restoring concurrently for worker pool (default: 16).
    ↪ Each worker restore one file at a time.
pool-size = 16

### The target database information
[db]
host = "127.0.0.1"
user = "root"
password = ""
port = 4000

### `sql_mode` of session level used to connect to the database when loading
    ↪  data. If `sql-mode` is not provided or set to "@DownstreamDefault",
    ↪ the global `sql_mode` for downstream is used.
### sql-mode = ""
### `max-allowed-packet` sets the maximum data packet allowed for database
    ↪ connection, which corresponds to the `max_allowed_packet` in system
    ↪ parameters. If it is set to 0, the global `max_allowed_packet` for
    ↪ downstream is used.
max-allowed-packet = 67108864
```

```
### The sharding replicating rules support wildcharacter.
### 1. The asterisk character (*, also called "star") matches zero or more
   ↪ characters,
###   for example, "doc*" matches "doc" and "document" but not "dodo";
###   asterisk character must be in the end of the wildcard word,
###   and there is only one asterisk in one wildcard word.
### 2. The question mark '?' matches exactly one character.
###   [[route-rules]]
###   pattern-schema = "shard_db_*"
###   pattern-table = "shard_table_*"
###   target-schema = "shard_db"
###   target-table = "shard_table"
```

4.16.7.4.3 Usage example

Command line parameter:

```
./bin/loader -d ./test -h 127.0.0.1 -u root -P 4000
```

Or use configuration file "config.toml":

```
./bin/loader -c=config.toml
```

4.16.7.5 FAQ

4.16.7.5.1 The scenario of replicating data from sharded tables

Loader supports importing data from sharded tables into one table within one database according to the route-rules. Before replicating, check the following items:

- Whether the sharding rules can be represented using the `route-rules` syntax.
- Whether the sharded tables contain monotone increasing primary keys, or whether there are conflicts in the unique indexes or the primary keys after the combination.

To combine tables, start the `route-rules` parameter in the configuration file of Loader:

- To use the table combination function, it is required to fill the `pattern-schema` and `target-schema`.
- If the `pattern-table` and `target-table` are NULL, the table name is not combined or converted.

```
[[route-rules]]
pattern-schema = "example_db"
pattern-table = "table_*"
target-schema = "example_db"
target-table = "table"
```

4.16.7.5.2 Error: `Try adjusting the `max_allowed_packet` variable`

The following error is reported during full data import:

```
packet for query is too large. Try adjusting the 'max_allowed_packet'
    ↪ variable
```

Reasons

- Both MySQL client and MySQL/TiDB Server have `max_allowed_packet` quotas. If any of the `max_allowed_packet` quotas is violated, the client receives a corresponding error message. Currently, the latest version of Loader and TiDB Server all have a default `max_allowed_packet` quota of `64M`.
  - Please use the latest version, or the latest stable version of the tool. See the download page.
- The full data import processing module in Loader or DM does not support splitting `dump sqls` files. This is because Mydumper has the simple code implementation, as shown in the code comment `/* Poor man's data dump code */`. To support correctly splitting `dump sqls` files, you need to implement a sound parser based on TiDB parser, but you will encounter the following issues:
  - A large amount of workload.
  - The task is difficult. It is not easy to ensure the correctness.
  - Significant performance reduction.

Solutions

- For the above reasons, it is recommended to use the `-s, --statement-size` option which Mydumper offers to control the size of `Insert Statement: Attempted size of INSERT statement in bytes, default 1000000`.

  According to the default configuration of `--statement-size`, the size of `Insert Statement` that Mydumper generates is as close as `1M`. This default configuration ensures that this error will not occur in most cases.

  Sometimes the following `WARN` log appears during the dump process. The `WARN` log itself does not affect the dump process but indicates that the dumped table might be a wide table.

```
Row bigger than statement_size for xxx
```

- If a single row of a wide table exceeds 64M, you need to modify and enable the following two configurations:
  - Execute `set @@global.max_allowed_packet=134217728` (`134217728 = 128M`) in TiDB Server.
  - Add `max-allowed-packet=128M` to db configuration in the Loader configuration file according to your situation, and then restart the progress or task.

4.16.8 TiDB Data Migration

TiDB Data Migration (DM) is an integrated data migration task management platform, which supports the full data migration and the incremental data replication from MySQL-compatible databases (such as MySQL, MariaDB, and Aurora MySQL) into TiDB. DM can help simplify the data migration process and reduce the operation cost of data migration.

4.16.8.1 DM versions

The stable versions of DM include v1.0, v2.0, and v5.3. It is recommended to use DM v5.3 (the latest stable version of DM) and not recommended to use v1.0 (the earliest stable version of DM).

Currently, the DM documentation is independent of the TiDB documentation. To access the DM documentation, click one of the following links:

- DM v5.3 documentation
- DM v2.0 documentation
- DM v1.0 documentation

Note:

- Since October 2021, DM's GitHub repository has been moved to pingcap/tiflow. If you see any issues with DM, submit your issue to the `pingcap/tiflow` repository for feedback.
- In earlier versions (v1.0 and v2.0), DM uses version numbers that are independent of TiDB. Since v5.3, DM uses the same version number as TiDB. The next version of DM v2.0 is DM v5.3. There are no compatibility changes from DM v2.0 to v5.3, and the upgrade process is no different from a normal upgrade, only an increase in version number.

4.16.8.2 Basic features

This section describes the basic data migration features provided by DM.

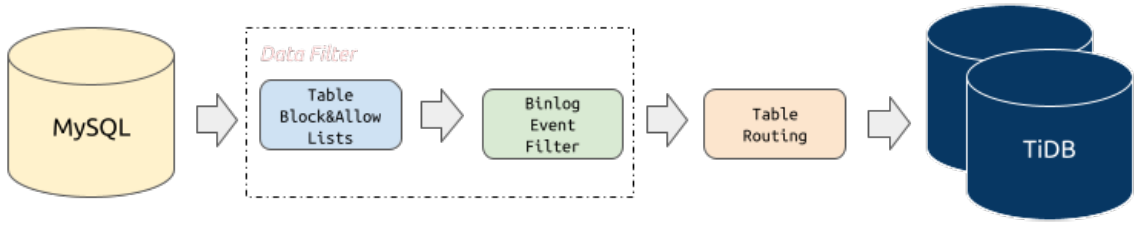

Figure 335: DM Core Features

4.16.8.2.1 Block and allow lists migration at the schema and table levels

The block and allow lists filtering rule is similar to the `replication-rules-db` ↪ `/replication-rules-table` feature of MySQL, which can be used to filter or replicate all operations of some databases only or some tables only.

4.16.8.2.2 Binlog event filtering

The binlog event filtering feature means that DM can filter certain types of SQL statements from certain tables in the source database. For example, you can filter all `INSERT` statements in the table `test.sbtest` or filter all `TRUNCATE TABLE` statements in the schema `test`.

4.16.8.2.3 Schema and table routing

The schema and table routing feature means that DM can migrate a certain table of the source database to the specified table in the downstream. For example, you can migrate the table structure and data from the table `test.sbtest1` in the source database to the table `test.sbtest2` in TiDB. This is also a core feature for merging and migrating sharded databases and tables.

4.16.8.3 Advanced features

4.16.8.3.1 Shard merge and migration

DM supports merging and migrating the original sharded instances and tables from the source databases into TiDB, but with some restrictions. For details, see Sharding DDL usage restrictions in the pessimistic mode and Sharding DDL usage restrictions in the optimistic mode.

4.16.8.3.2 Optimization for third-party online-schema-change tools in the migration process

In the MySQL ecosystem, tools such as `gh-ost` and `pt-osc` are widely used. DM provides support for these tools to avoid migrating unnecessary intermediate data. For details, see Online DDL Tools

4.16.8.3.3 Filter certain row changes using SQL expressions

In the phase of incremental replication, DM supports the configuration of SQL expressions to filter out certain row changes, which lets you replicate the data with a greater granularity. For more information, refer to Filter Certain Row Changes Using SQL Expressions.

4.16.8.4 Usage restrictions

Before using the DM tool, note the following restrictions:

- Database version requirements
  - MySQL version > 5.5
  - MariaDB version >= 10.1.2

> **Note:**
>
> If there is a primary-secondary migration structure between the upstream MySQL/MariaDB servers, then choose the following version.
>
> - MySQL version > 5.7.1
> - MariaDB version >= 10.1.3

> **Warning:**
>
> Migrating data from MySQL 8.0 to TiDB using DM is an experimental feature (introduced since DM v2.0). It is **NOT** recommended that you use it in a production environment.

- DDL syntax compatibility

  - Currently, TiDB is not compatible with all the DDL statements that MySQL supports. Because DM uses the TiDB parser to process DDL statements, it only supports the DDL syntax supported by the TiDB parser. For details, see MySQL Compatibility.
  - DM reports an error when it encounters an incompatible DDL statement. To solve this error, you need to manually handle it using dmctl, either skipping this DDL statement or replacing it with a specified DDL statement(s). For details, see Skip or replace abnormal SQL statements.

- Sharding merge with conflicts

  - If conflict exists between sharded tables, solve the conflict by referring to handling conflicts of auto-increment primary key. Otherwise, data migration is not supported. Conflicting data can cover each other and cause data loss.
  - For other sharding DDL migration restrictions, see Sharding DDL usage restrictions in the pessimistic mode and Sharding DDL usage restrictions in the optimistic mode.

- Switch of MySQL instances for data sources

  When DM-worker connects the upstream MySQL instance via a virtual IP (VIP), if you switch the VIP connection to another MySQL instance, DM might connect to the new and old MySQL instances at the same time in different connections. In this situation, the binlog migrated to DM is not consistent with other upstream status that DM receives, causing unpredictable anomalies and even data damage. To make necessary changes to DM manually, see Switch DM-worker connection via virtual IP.

4.16.9 TiDB Lightning

4.16.9.1 TiDB Lightning Overview

TiDB Lightning is a tool used for fast full import of large amounts of data into a TiDB cluster. You can download TiDB Lightning from here.

Currently, TiDB Lightning supports reading SQL dump exported via Mydumper or CSV data source. You can use it in the following two scenarios:

- Importing **large amounts** of **new** data **quickly**
- Restore all backup data

4.16.9.1.1 TiDB Lightning architecture

The TiDB Lightning tool set consists of two components:

- **`tidb-lightning`** (the "front end") reads the data source and imports the database structure into the TiDB cluster, and also transforms the data into Key-Value (KV) pairs and sends them to `tikv-importer`.

- **`tikv-importer`** (the "back end") combines and sorts the KV pairs and then imports these sorted pairs as a whole into the TiKV cluster.

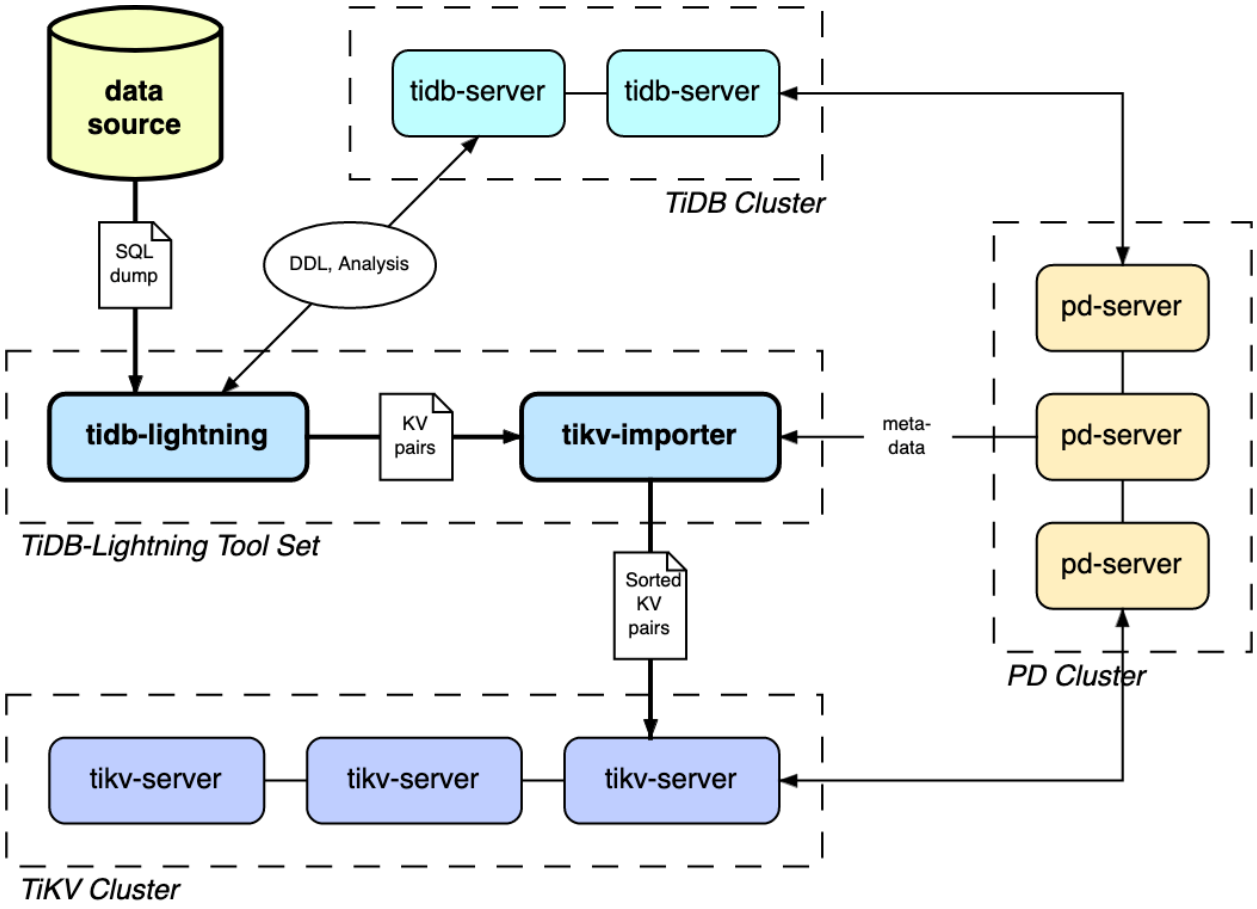

Figure 336: Architecture of TiDB Lightning tool set

The complete import process is as follows:

1. Before importing, `tidb-lightning` switches the TiKV cluster to "import mode", which optimizes the cluster for writing and disables automatic compaction.
2. `tidb-lightning` creates the skeleton of all tables from the data source.
3. Each table is split into multiple continuous *batches*, so that data from a huge table (200 GB+) can be delivered incrementally.
4. For each batch, `tidb-lightning` informs `tikv-importer` via gRPC to create *engine files* to store KV pairs. `tidb-lightning` then reads the data source in parallel, transforms each row into KV pairs according to the TiDB rules, and sends them to `tikv-importer`'s engine files.

5. Once a complete engine file is written, `tikv-importer` divides and schedules these data and imports them into the target TiKV cluster.

   There are two kinds of engine files: *data engines* and *index engines*, each corresponding to two kinds of KV pairs: the row data and secondary indices. Normally, the row data are entirely sorted in the data source, while the secondary indices are out of order. Because of this, the data engines are uploaded as soon as a batch is completed, while the index engines are imported only after all batches of the entire table are encoded.

6. After all engines associated to a table are imported, `tidb-lightning` performs a checksum comparison between the local data source and those calculated from the cluster, to ensure there is no data corruption in the process; tells TiDB to `ANALYZE` all imported tables, to prepare for optimal query planning; and adjusts the `AUTO_INCREMENT` value so future insertions will not cause conflict.

   The auto-increment ID of a table is computed by the estimated *upper bound* of the number of rows, which is proportional to the total file size of the data files of the table. Therefore, the final auto-increment ID is often much larger than the actual number of rows. This is expected since in TiDB auto-increment is not necessarily allocated sequentially.

7. Finally, `tidb-lightning` switches the TiKV cluster back to "normal mode", so the cluster resumes normal services.

TiDB Lightning also supports using TiDB instead of Importer as the backend. In this configuration, `tidb-lightning` transforms data into SQL `INSERT` statements and directly executes them on the target cluster, similar to Loader. See TiDB Lightning TiDB-backend for details.

4.16.9.2 TiDB Lightning Deployment

This document describes the hardware requirements of TiDB Lightning using the default Importer-backend, and how to deploy it using TiDB Ansible or manually.

If you do not want the TiDB services to be impacted, read TiDB Lightning TiDB-backend for the changes to the deployment steps.

4.16.9.2.1 Notes

Before starting TiDB Lightning, note that:

- During the import process, the cluster cannot provide normal services.
- If `tidb-lightning` crashes, the cluster is left in "import mode". Forgetting to switch back to "normal mode" can lead to a high amount of uncompacted data on the TiKV cluster, and cause abnormally high CPU usage and stall. You can manually switch the cluster back to "normal mode" via the `tidb-lightning-ctl` tool:

```
bin/tidb-lightning-ctl -switch-mode=normal
```

- TiDB Lightning is required to have the following privileges in the downstream TiDB:

Privilege	Scope
SELECT	Tables
INSERT	Tables
UPDATE	Tables
DELETE	Tables
CREATE	Databases, tables
DROP	Databases, tables
ALTER	Tables

  If the `checksum` configuration item of TiDB Lightning is set to `true`, then the admin user privileges in the downstream TiDB need to be granted to TiDB Lightning.

4.16.9.2.2 Hardware requirements

`tidb-lightning` and `tikv-importer` are both resource-intensive programs. It is recommended to deploy them into two separate machines.

To achieve the best performance, it is recommended to use the following hardware configuration:

- `tidb-lightning`:
  - 32+ logical cores CPU
  - An SSD large enough to store the entire data source, preferring higher read speed
  - 10 Gigabit network card (capable of transferring at 300 MB/s)
  - `tidb-lightning` fully consumes all CPU cores when running, and deploying on a dedicated machine is highly recommended. If not possible, `tidb-lightning` could be deployed together with other components like `tidb-server`, and the CPU usage could be limited via the `region-concurrency` setting.
- `tikv-importer`:
  - 32+ logical cores CPU
  - 40 GB+ memory
  - 1 TB+ SSD, preferring higher IOPS (8000 is recommended)
    - The disk should be larger than the total size of the top N tables, where N = max(index-concurrency, table-concurrency).
  - 10 Gigabit network card (capable of transferring at 300 MB/s)
  - `tikv-importer` fully consumes all CPU, disk I/O and network bandwidth when running, and deploying on a dedicated machine is strongly recommended.

If you have sufficient machines, you can deploy multiple Lightning/Importer servers, with each working on a distinct set of tables, to import the data in parallel.

Note:

- `tidb-lightning` is a CPU intensive program. In an environment with mixed components, the resources allocated to `tidb-lightning` must be limited. Otherwise, other components might not be able to run. It is recommended to set the `region-concurrency` to 75% of CPU logical cores. For instance, if the CPU has 32 logical cores, you can set the `region-concurrency` to 24.
- `tikv-importer` stores intermediate data on the RAM to speed up the import process. The typical memory usage can be calculated by using **(`max-open-engines` × `write-buffer-size` × 2) + (`num-import-jobs` × `region-split-size` × 2)**. If the speed of writing to disk is slow, the memory usage could be even higher due to buffering.

Additionally, the target TiKV cluster should have enough space to absorb the new data. Besides the standard requirements, the total free space of the target TiKV cluster should be larger than **Size of data source × Number of replicas × 2**.

With the default replica count of 3, this means the total free space should be at least 6 times the size of data source.

4.16.9.2.3 Export data

Use the `mydumper` tool to export data from MySQL by using the following command:

```
./bin/mydumper -h 127.0.0.1 -P 3306 -u root -t 16 -F 256 -B test -T t1,t2
    ↪ --skip-tz-utc -o /data/my_database/
```

In this command,

- `-B test`: means the data is exported from the `test` database.
- `-T t1,t2`: means only the `t1` and `t2` tables are exported.
- `-t 16`: means 16 threads are used to export the data.
- `-F 256`: means a table is partitioned into chunks and one chunk is 256 MB.
- `--skip-tz-utc`: the purpose of adding this parameter is to ignore the inconsistency of time zone setting between MySQL and the data exporting machine, and to disable automatic conversion.

If the data source consists of CSV files, see CSV support for configuration.

4.16.9.2.4 Deploy TiDB Lightning

This section describes two deployment methods of TiDB Lightning:

- Deploy TiDB Lightning using TiDB Ansible
- Deploy TiDB Lightning manually

Deploy TiDB Lightning using TiDB Ansible

You can deploy TiDB Lightning using TiDB Ansible together with the deployment of the TiDB cluster itself using TiDB Ansible.

1. Edit `inventory.ini` to add the addresses of the `tidb-lightning` and `tikv-importer` servers.

```
...

[importer_server]
192.168.20.9

[lightning_server]
192.168.20.10

...
```

2. Configure these tools by editing the settings under `group_vars/*.yml`.

    - `group_vars/all.yml`

```
...
# The listening port of tikv-importer. Should be open to the tidb-
    ↪ lightning server.
tikv_importer_port: 8287
...
```

    - `group_vars/lightning_server.yml`

```
---
dummy:

# The listening port for metrics gathering. Should be open to the
    ↪ monitoring servers.
tidb_lightning_pprof_port: 8289

# The file path that tidb-lightning reads the data source (Mydumper
    ↪  SQL dump or CSV) from.
data_source_dir: "{{ deploy_dir }}/mydumper"
```

- `group_vars/importer_server.yml`

```
---
dummy:

# The file path to store engine files. Should reside on a partition
    ↪ with a large capacity.
import_dir: "{{ deploy_dir }}/data.import"
```

3. Deploy the cluster.

```
ansible-playbook bootstrap.yml
ansible-playbook deploy.yml
```

4. Mount the data source to the path specified in the `data_source_dir` setting.

5. Log in to the `tikv-importer` server, and manually run the following command to start Importer.

```
scripts/start_importer.sh
```

6. Log in to the `tidb-lightning` server, and manually run the following command to start Lightning and import the data into the TiDB cluster.

```
scripts/start_lightning.sh
```

7. After completion, run `scripts/stop_importer.sh` on the `tikv-importer` server to stop Importer.

Deploy TiDB Lightning manually

Step 1: Deploy a TiDB cluster

Before importing data, you need to have a deployed TiDB cluster, with the cluster version 2.0.9 or above. It is highly recommended to use the latest version.

You can find deployment instructions in TiDB Quick Start Guide.

Step 2: Download the TiDB Lightning installation package

Refer to the TiDB enterprise tools download page to download the TiDB Lightning package (choose the same version as that of the TiDB cluster).

Step 3: Start `tikv-importer`

1. Upload `bin/tikv-importer` from the installation package.

2. Configure `tikv-importer.toml`.

```
# TiKV Importer configuration file template

# Log file
log-file = "tikv-importer.log"
# Log level: trace, debug, info, warn, error, off.
log-level = "info"

[server]
# The listening address of tikv-importer. tidb-lightning needs to
   ↪ connect to
# this address to write data.
addr = "192.168.20.10:8287"

[metric]
# The Prometheus client push job name.
job = "tikv-importer"
# The Prometheus client push interval.
interval = "15s"
# The Prometheus Pushgateway address.
address = ""

[import]
# The directory to store engine files.
import-dir = "/mnt/ssd/data.import/"
```

The above only shows the essential settings. See the Configuration section for the full list of settings.

3. Run `tikv-importer`.

```
nohup ./tikv-importer -C tikv-importer.toml > nohup.out &
```

Step 4: Start `tidb-lightning`

1. Upload `bin/tidb-lightning` and `bin/tidb-lightning-ctl` from the tool set.
2. Mount the data source onto the same machine.
3. Configure `tidb-lightning.toml`. For configurations that do not appear in the template below, TiDB Lightning writes a configuration error to the log file and exits.

```
[lightning]
# The concurrency number of data. It is set to the number of logical
   ↪ CPU
```

```
# cores by default. When deploying together with other components, you
    ↪ can
# set it to 75% of the size of logical CPU cores to limit the CPU usage
    ↪ .
# region-concurrency =

# Logging
level = "info"
file = "tidb-lightning.log"

[tikv-importer]
# Delivery backend, can be "importer" or "tidb".
# backend = "importer"
# The listening address of tikv-importer when backend is "importer".
    ↪ Change it to the actual address.
addr = "172.16.31.10:8287"

[mydumper]
# mydumper local source data directory
data-source-dir = "/data/my_database"

[tidb]
# Configuration of any TiDB server from the cluster
host = "172.16.31.1"
port = 4000
user = "root"
password = ""
# Table schema information is fetched from TiDB via this status-port.
status-port = 10080
```

The above only shows the essential settings. See the Configuration section for the full list of settings.

4. Run `tidb-lightning`.

```
nohup ./tidb-lightning -config tidb-lightning.toml > nohup.out &
```

4.16.9.2.5 Upgrading TiDB Lightning

You can upgrade TiDB Lightning by replacing the binaries alone. No further configuration is needed. See FAQ for the detailed instructions of restarting TiDB Lightning.

If an import task is running, we recommend you to wait until it finishes before upgrading TiDB Lightning. Otherwise, there might be chances that you need to reimport from scratch, because there is no guarantee that checkpoints work across versions.

4.16.9.3 TiDB Lightning Configuration

This document provides samples for global configuration, task configuration, and TiKV Importer configuration in TiDB Lightning, and describes the usage of command-line parameters.

4.16.9.3.1 Configuration files

TiDB Lightning has two configuration classes: "global" and "task", and they have compatible structures. Their distinction arises only when the server mode is enabled. When server mode is disabled (the default), TiDB Lightning will only execute one task, and the same configuration file is used for both global and task configurations.

TiDB Lightning (Global)

```
###### tidb-lightning global configuration

[lightning]
#### The HTTP port for the web interface and Prometheus metrics pulling (0
    ↪ to disable).
status-addr = ':8289'

#### Toggle server mode and use of the web interface.
#### See the "TiDB Lightning Web Interface" section for details.
server-mode = false

#### Logging
level = "info"
file = "tidb-lightning.log"
max-size = 128 # MB
max-days = 28
max-backups = 14
```

TiDB Lightning (Task)

```
###### tidb-lightning task configuration

[lightning]
#### Checks whether the cluster satisfies the minimum requirement before
    ↪ starting.
#check-requirements = true

#### The maximum number of engines to be opened concurrently.
#### Each table is split into one "index engine" to store indices, and
    ↪ multiple
#### "data engines" to store row data. These settings control the maximum
#### concurrent number for each type of engines.
```

```
#### These values affect the memory and disk usage of tikv-importer.
#### The sum of these two values must not exceed the max-open-engines
   ↪ setting
#### for tikv-importer.
index-concurrency = 2
table-concurrency = 6

#### The concurrency number of data. It is set to the number of logical CPU
#### cores by default. When deploying together with other components, you
   ↪ can
#### set it to 75% of the size of logical CPU cores to limit the CPU usage.
#region-concurrency =

#### The maximum I/O concurrency. Excessive I/O concurrency causes an
   ↪ increase in
#### I/O latency because the disk's internal buffer is frequently refreshed,
#### which causes the cache miss and slows down the read speed. Depending on
   ↪  the storage
#### medium, this value might need to be adjusted for optimal performance.
io-concurrency = 5

[checkpoint]
#### Whether to enable checkpoints.
#### While importing data, TiDB Lightning records which tables have been
   ↪ imported, so
#### even if Lightning or another component crashes, you can start from a
   ↪ known
#### good state instead of redoing everything.
enable = true
#### The schema name (database name) to store the checkpoints.
schema = "tidb_lightning_checkpoint"
#### Where to store the checkpoints.
####  - file: store as a local file.
####  - mysql: store into a remote MySQL-compatible database
driver = "file"
#### The data source name (DSN) indicating the location of the checkpoint
   ↪ storage.
#### For the "file" driver, the DSN is a path. If the path is not specified,
   ↪  Lightning would
#### default to "/tmp/CHECKPOINT_SCHEMA.pb".
#### For the "mysql" driver, the DSN is a URL in the form of "USER:PASS@tcp(
   ↪ HOST:PORT)/".
#### If the URL is not specified, the TiDB server from the [tidb] section is
   ↪  used to
#### store the checkpoints. You should specify a different MySQL-compatible
```

```
#### database server to reduce the load of the target TiDB cluster.
#dsn = "/tmp/tidb_lightning_checkpoint.pb"
#### Whether to keep the checkpoints after all data are imported. If false,
    ↪ the
#### checkpoints will be deleted. Keeping the checkpoints can aid debugging
    ↪ but
#### will leak metadata about the data source.
#keep-after-success = false

[tikv-importer]
#### The listening address of tikv-importer. Change it to the actual address
    ↪ .
addr = "172.16.31.10:8287"

[mydumper]
#### Block size for file reading. Keep it longer than the longest string of
#### the data source.
read-block-size = 65536 # Byte (default = 64 KB)

#### Minimum size (in terms of source data file) of each batch of import.
#### TiDB Lightning splits a large table into multiple data engine files
    ↪ according to this size.
batch-size = 107_374_182_400 # Byte (default = 100 GB)

#### The engine file needs to be imported sequentially. Due to parallel
    ↪ processing,
#### multiple data engines will be imported at nearly the same time, and
    ↪ this
#### creates a queue and wastes resources. Therefore, Lightning slightly
#### increases the size of the first few batches to properly distribute
#### resources. The scale up factor is controlled by this parameter, which
#### expresses the ratio of duration between the "import" and "write" steps
#### with full concurrency. This can be calculated by using the ratio
#### (import duration/write duration) of a single table of size around 1 GB.
#### The exact timing can be found in the log. If "import" is faster, the
    ↪ batch
#### size variance is smaller, and a ratio of zero means a uniform batch
    ↪ size.
#### This value should be in the range (0 <= batch-import-ratio < 1).
batch-import-ratio = 0.75

#### mydumper local source data directory.
data-source-dir = "/data/my_database"
#### If no-schema is set to true, tidb-lightning assumes that the table
    ↪ skeletons
```

```
#### already exist on the target TiDB cluster, and will not execute the `
   ↪ CREATE
#### TABLE` statements.
no-schema = false
#### The character set of the schema files, containing CREATE TABLE
   ↪ statements;
#### only supports one of:
#### - utf8mb4: the schema files must be encoded as UTF-8, otherwise
   ↪ Lightning
####           will emit errors
#### - gb18030: the schema files must be encoded as GB-18030, otherwise
####           Lightning will emit errors
#### - auto:   (default) automatically detects whether the schema is UTF-8
   ↪ or
####           GB-18030. An error is reported if the encoding is neither.
#### - binary: do not try to decode the schema files
#### Note that the *data* files are always parsed as binary regardless of
#### schema encoding.
character-set = "auto"

#### Only import tables if these wildcard rules are matched. See the
   ↪ corresponding section for details.
filter = ['*.*']

#### Configures how CSV files are parsed.
[mydumper.csv]
#### Separator between fields, should be an ASCII character.
separator = ','
#### Quoting delimiter, can either be an ASCII character or empty string.
delimiter = '"'
#### Whether the CSV files contain a header.
#### If `header` is true, the first line will be skipped.
header = true
#### Whether the CSV contains any NULL value.
#### If `not-null` is true, all columns from CSV cannot be NULL.
not-null = false
#### When `not-null` is false (that is, CSV can contain NULL),
#### fields equal to this value will be treated as NULL.
null = '\N'
#### Whether to interpret backslash escapes inside fields.
backslash-escape = true
#### If a line ends with a separator, remove it.
trim-last-separator = false

[tidb]
```

```
#### Configuration of any TiDB server from the cluster.
host = "172.16.31.1"
port = 4000
user = "root"
password = ""
#### Table schema information is fetched from TiDB via this status-port.
status-port = 10080
#### Address of any PD server from the cluster.
pd-addr = "172.16.31.4:2379"
#### tidb-lightning imports TiDB as a library and generates some logs itself
    ↪ .
#### This setting controls the log level of the TiDB library.
log-level = "error"

#### Sets the TiDB session variable to speed up the Checksum and Analyze
    ↪ operations.
#### See https://pingcap.com/docs/v3.0/reference/performance/statistics/#
    ↪ control-analyze-concurrency
#### for the meaning of each setting
build-stats-concurrency = 20
distsql-scan-concurrency = 100
index-serial-scan-concurrency = 20
checksum-table-concurrency = 16

#### The default SQL mode used to parse and execute the SQL statements.
sql-mode = "STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION"

#### When data importing is complete, tidb-lightning can automatically
    ↪ perform
#### the Checksum, Compact and Analyze operations. It is recommended to
    ↪ leave
#### these as true in the production environment.
#### The execution order: Checksum -> Analyze
[post-restore]
#### Performs `ADMIN CHECKSUM TABLE <table>` for each table to verify data
    ↪ integrity.
checksum = true
#### If the value is set to `true`, a level-1 compaction is performed
#### every time a table is imported.
#### The default value is `false`.
level-1-compact = false
#### If the value is set to `true`, a full compaction on the whole
#### TiKV cluster is performed at the end of the import.
#### The default value is `false`.
compact = false
```

```
#### Performs `ANALYZE TABLE <table>` for each table.
analyze = true

#### Configures the background periodic actions.
#### Supported units: h (hour), m (minute), s (second).
[cron]
#### Duration between which Lightning automatically refreshes the import
    ↪ mode
#### status. Should be shorter than the corresponding TiKV setting.
switch-mode = "5m"
#### Duration between which an import progress is printed to the log.
log-progress = "5m"
```

TiKV Importer

```
#### TiKV Importer configuration file template.

#### Log file.
log-file = "tikv-importer.log"
#### Log level: trace, debug, info, warn, error, off.
log-level = "info"

[server]
#### The listening address of tikv-importer. tidb-lightning needs to connect
    ↪ to
#### this address to write data.
addr = "192.168.20.10:8287"
#### Size of the thread pool for the gRPC server.
grpc-concurrency = 16

[metric]
#### The Prometheus client push job name.
job = "tikv-importer"
#### The Prometheus client push interval.
interval = "15s"
#### The Prometheus Pushgateway address.
address = ""

[rocksdb]
#### The maximum number of concurrent background jobs.
max-background-jobs = 32

[rocksdb.defaultcf]
#### Amount of data to build up in memory before flushing data to the disk.
write-buffer-size = "1GB"
```

```
#### The maximum number of write buffers that are built up in memory.
max-write-buffer-number = 8

#### The compression algorithms used in different levels.
#### The algorithm at level-0 is used to compress KV data.
#### The algorithm at level-6 is used to compress SST files.
#### The algorithms at level-1 to level-5 are unused for now.
compression-per-level = ["lz4", "no", "no", "no", "no", "no", "lz4"]

[rocksdb.writecf]
#### (same as above)
compression-per-level = ["lz4", "no", "no", "no", "no", "no", "lz4"]

[import]
#### The directory to store engine files.
import-dir = "/mnt/ssd/data.import/"
#### Number of threads to handle RPC requests.
num-threads = 16
#### Number of concurrent import jobs.
num-import-jobs = 24
#### Maximum duration to prepare Regions.
#max-prepare-duration = "5m"
#### Split Regions into this size according to the importing data.
#region-split-size = "512MB"
#### Stream channel window size. The stream will be blocked on channel full.
#stream-channel-window = 128
#### Maximum number of open engines.
max-open-engines = 8
#### Maximum upload speed (bytes per second) from Importer to TiKV.
#### upload-speed-limit = "512MB"
#### Minimum ratio of available space on the target store: `
   ↪ store_available_space`/`store_capacity`.
#### Importer pauses uploading SST if the availability ratio of the target
   ↪ store is less than this
#### value, to allow enough time for PD to balance Regions.
min-available-ratio = 0.05
```

4.16.9.3.2 Command line parameters

Usage of `tidb-lightning`

Parameter	Explanation	Corresponding setting
–config *file*	Reads global configuration from *file*. If not specified, the default configuration would be used.	
-V	Prints program version	
-d *directory*	Directory of the data dump to read from	`mydumper .` `data -` `source -` `dir`

Parameter	Explanation	Corresponding setting
-L *level*	Log level: debug, info, warn, error, fatal (default = info)	`lightning.log.level`
-f *rule*	Table filter rules (can be specified multiple times)	`mydumper.filter`
–log-file *file*	Log file path	`lightning.log.file`
–status-addr *ip:port*	Listening address of the TiDB Lightning server	`lightning.status.port`
–importer *host:port*	Address of TiKV Importer	`tikv-importer.addr`

Parameter	Explanation	Corresponding setting
–pd-urls *host:port*	PD endpoint address	`tidb.pd-addr`
–tidb-host *host*	TiDB server host	`tidb.host`
–tidb-port *port*	TiDB server port (default = 4000)	`tidb.port`
–tidb-status *port*	TiDB status port (default = 10080)	`tidb.status-port`
–tidb-user *user*	User name to connect to TiDB	`tidb.user`
–tidb-password *password*	Password to connect to TiDB	`tidb.password`

If a command line parameter and the corresponding setting in the configuration file are both provided, the command line parameter will be used. For example, running `./tidb-lightning -L debug --config cfg.toml` would always set the log level to "debug" regardless of the content of `cfg.toml`.

4.16.9.3.3 Usage of `tidb-lightning-ctl`

This tool can execute various actions given one of the following parameters:

Parameter	Explanation
–compact	Performs a full compaction
–switch-mode *mode*	Switches every TiKV store to the given mode: normal, import
–import-engine *uuid*	Imports the closed engine file from TiKV Importer into the TiKV cluster
–cleanup-engine *uuid*	Deletes the engine file from TiKV Importer

Parameter	Explanation
–checkpoint-dump *folder*	Dumps current checkpoint as CSVs into the folder
–checkpoint-error-destroy *table-name*	Removes the checkpoint and drops the table if it caused error
–checkpoint-error-ignore *table-name*	Ignores any error recorded in the checkpoint involving the given table
–checkpoint-remove *table-name*	Unconditionally removes the checkpoint of the table

The *tablename* must either be a qualified table name in the form `` `db`.`tbl` `` (including the backquotes), or the keyword "all".

Additionally, all parameters of `tidb-lightning` described in the section above are valid in `tidb-lightning-ctl`.

4.16.9.3.4 Usage of `tikv-importer`

Parameter	Explanation	Corresponding setting
-C, –config *file*	Reads configuration from *file*. If not specified, the default configuration would be used.	
-V, –version	Prints program version	
-A, –addr *ip:port*	Listening address of the TiKV Importer server	`server.addr`

Parameter	Explanation	Corresponding setting
–import-dir *dir*	Stores engine files in this directory	`import.import-dir`
–log-level *level*	Log level: trace, debug, info, warn, error, off	`log.level`
–log-file *file*	Log file path	`log.file`

4.16.9.4 TiDB Lightning Checkpoints

Importing a large database usually takes hours or days, and if such long running processes spuriously crashes, it can be very time-wasting to redo the previously completed tasks. To solve this, Lightning uses *checkpoints* to store the import progress, so that `tidb-lightning` continues importing from where it lefts off after restarting.

This document describes how to enable, configure, store, and control *checkpoints*.

4.16.9.4.1 Enable and configure checkpoints

```
[checkpoint]
#### Whether to enable checkpoints.
#### While importing data, Lightning records which tables have been imported
   ↪ , so
#### even if Lightning or some other component crashes, you can start from a
   ↪  known
#### good state instead of redoing everything.
enable = true

#### Where to store the checkpoints.
#### - file: store as a local file (requires v2.1.1 or later)
#### - mysql: store into a remote MySQL-compatible database
```

```
driver = "file"

#### The schema name (database name) to store the checkpoints
#### Enabled only when `driver = "mysql"`.
#### schema = "tidb_lightning_checkpoint"

#### The data source name (DSN) indicating the location of the checkpoint
    ↪ storage.
#### # For the "file" driver, the DSN is a path. If the path is not
    ↪ specified, Lightning would
#### default to "/tmp/CHECKPOINT_SCHEMA.pb".
#### # For the "mysql" driver, the DSN is a URL in the form of "USER:
    ↪ PASS@tcp(HOST:PORT)/".
#### If the URL is not specified, the TiDB server from the [tidb] section is
    ↪  used to
#### store the checkpoints. You should specify a different MySQL-compatible
#### database server to reduce the load of the target TiDB cluster.
#dsn = "/tmp/tidb_lightning_checkpoint.pb"

#### Whether to keep the checkpoints after all data are imported. If false,
    ↪ the
#### checkpoints are deleted. Keeping the checkpoints can aid debugging but
#### might leak metadata about the data source.
#### keep-after-success = false
```

4.16.9.4.2 Checkpoints storage

TiDB Lightning supports two kinds of checkpoint storage: a local file or a remote MySQL-compatible database.

- With `driver = "file"`, checkpoints are stored in a local file at the path given by the `dsn` setting. Checkpoints are updated rapidly, so we highly recommend placing the checkpoint file on a drive with very high write endurance, such as a RAM disk.
- With `driver = "mysql"`, checkpoints can be saved in any databases compatible with MySQL 5.7 or later, including MariaDB and TiDB. By default, the checkpoints are saved in the target database.

While using the target database as the checkpoints storage, Lightning is importing large amounts of data at the same time. This puts extra stress on the target database and sometimes leads to communication timeout. Therefore, **it is strongly recommended to install a temporary MySQL server to store these checkpoints**. This server can be installed on the same host as `tidb-lightning` and can be uninstalled after the importer progress is completed.

4.16.9.4.3 Checkpoints control

If `tidb-lightning` exits abnormally due to unrecoverable errors (for example, data corruption), it refuses to reuse the checkpoints until the errors are resolved. This is to prevent worsening the situation. The checkpoint errors can be resolved using the `tidb-lightning-ctl` program.

`--checkpoint-error-destroy`

```
tidb-lightning-ctl --checkpoint-error-destroy='`schema`.`table`'
```

This option allows you to restart importing the table from scratch. The schema and table names must be quoted with backquotes and are case-sensitive.

- If importing the table `` `schema`.`table` `` failed previously, this option executes the following operations:

    1. DROPs the table `` `schema`.`table` `` from the target database, which means removing all imported data.
    2. Resets the checkpoints record of this table to be "not yet started".

- If there is no errors involving the table `` `schema`.`table` ``, this operation does nothing.

It is the same as applying the above on every table. This is the most convenient, safe and conservative solution to fix the checkpoint error problem:

```
tidb-lightning-ctl --checkpoint-error-destroy=all
```

`--checkpoint-error-ignore`

```
tidb-lightning-ctl --checkpoint-error-ignore='`schema`.`table`'
tidb-lightning-ctl --checkpoint-error-ignore=all
```

If importing the table `` `schema`.`table` `` failed previously, this clears the error status as if nothing ever happened. The `all` variant applies this operation to all tables.

> **Note:**
>
> Use this option only when you are sure that the error can indeed be ignored. If not, some imported data can be lost. The only safety net is the final "checksum" check, and thus you need to keep the "checksum" option always enabled when using `--checkpoint-error-ignore`.

`--checkpoint-remove`

```
tidb-lightning-ctl --checkpoint-remove='`schema`.`table`'
tidb-lightning-ctl --checkpoint-remove=all
```

This option simply removes all checkpoint information about one table or all tables, regardless of their status.

`--checkpoint-dump`

```
tidb-lightning-ctl --checkpoint-dump=output/directory
```

This option dumps the content of the checkpoint into the given directory, which is mainly used for debugging by the technical staff. This option is only enabled when `driver = "mysql ↪ "`.

4.16.9.5 Table Filter

The TiDB ecosystem tools operate on all the databases by default, but oftentimes only a subset is needed. For example, you only want to work with the schemas in the form of `foo*` and `bar*` and nothing else.

Several TiDB ecosystem tools share a common filter syntax to define subsets. This document describes how to use the table filter feature.

4.16.9.5.1 Usage

CLI

Table filters can be applied to the tools using multiple `-f` or `--filter` command line parameters. Each filter is in the form of `db.table`, where each part can be a wildcard (further explained in the next section). The following lists the example usage in each tool.

- Dumpling:

```
./dumpling -f 'foo*.*' -f 'bar*.*' -P 3306 -o /tmp/data/
#          ^~~~~~~~~~~~~~~~~~~~~~
```

- Lightning:

```
./tidb-lightning -f 'foo*.*' -f 'bar*.*' -d /tmp/data/ --backend tidb
#                ^~~~~~~~~~~~~~~~~~~~~~
```

TOML configuration files

Table filters in TOML files are specified as array of strings. The following lists the example usage in each tool.

- Lightning:

```
[mydumper]
filter = ['foo*.*', 'bar*.*']
```

4.16.9.5.2 Syntax

Plain table names

Each table filter rule consists of a "schema pattern" and a "table pattern", separated by a dot (`.`). Tables whose fully-qualified name matches the rules are accepted.

```
db1.tbl1
db2.tbl2
db3.tbl3
```

A plain name must only consist of valid identifier characters, such as:

- digits (`0` to `9`)
- letters (`a` to `z`, `A` to `Z`)
- `$`
- `_`
- non ASCII characters (U+0080 to U+10FFFF)

All other ASCII characters are reserved. Some punctuations have special meanings, as described in the next section.

Wildcards

Each part of the name can be a wildcard symbol described in fnmatch(3):

- `*` — matches zero or more characters
- `?` — matches one character
- `[a-z]` — matches one character between "a" and "z" inclusively
- `[!a-z]` — matches one character except "a" to "z".

```
db[0-9].tbl[0-9a-f][0-9a-f]
data.*
*.backup_*
```

"Character" here means a Unicode code point, such as:

- U+00E9 (é) is 1 character.
- U+0065 U+0301 (é) are 2 characters.
- U+1F926 U+1F3FF U+200D U+2640 U+FE0F ( ) are 5 characters.

File import

To import a file as the filter rule, include an `@` at the beginning of the rule to specify the file name. The table filter parser treats each line of the imported file as additional filter rules.

For example, if a file `config/filter.txt` has the following content:

```
employees.*
*.WorkOrder
```

the following two invocations are equivalent:

```
./dumpling -f '@config/filter.txt'
./dumpling -f 'employees.*' -f '*.WorkOrder'
```

A filter file cannot further import another file.

Comments and blank lines

Inside a filter file, leading and trailing white-spaces of every line are trimmed. Furthermore, blank lines (empty strings) are ignored.

A leading `#` marks a comment and is ignored. `#` not at start of line is considered syntax error.

```
#### this line is a comment
db.table  # but this part is not comment and may cause error
```

Exclusion

An `!` at the beginning of the rule means the pattern after it is used to exclude tables from being processed. This effectively turns the filter into a block list.

```
*.*
#^ note: must add the *.* to include all tables first
!*.Password
!employees.salaries
```

Escape character

To turn a special character into an identifier character, precede it with a backslash `\`.

```
db\.with\.dots.*
```

For simplicity and future compatibility, the following sequences are prohibited:

- `\` at the end of the line after trimming whitespaces (use `[ ]` to match a literal whitespace at the end).
- `\` followed by any ASCII alphanumeric character (`[0-9a-zA-Z]`). In particular, C-like escape sequences like `\0`, `\r`, `\n` and `\t` currently are meaningless.

Quoted identifier

Besides `\`, special characters can also be suppressed by quoting using `"` or `` ` ``.

```
"db.with.dots"."tbl\1"
`db.with.dots`.`tbl\2`
```

The quotation mark can be included within an identifier by doubling itself.

```
"foo""bar".`foo``bar`
#### equivalent to:
foo\"bar.foo\`bar
```

Quoted identifiers cannot span multiple lines.

It is invalid to partially quote an identifier:

```
"this is "invalid*.*
```

Regular expression

In case very complex rules are needed, each pattern can be written as a regular expression delimited with `/`:

```
/^db\d{2,}$/./^tbl\d{2,}$/
```

These regular expressions use the Go dialect. The pattern is matched if the identifier contains a substring matching the regular expression. For instance, `/b/` matches `db01`.

> **Note:**
>
> Every `/` in the regular expression must be escaped as `\/`, including inside `[···]`. You cannot place an unescaped `/` between `\Q···\E`.

4.16.9.5.3 Multiple rules

When a table name matches none of the rules in the filter list, the default behavior is to ignore such unmatched tables.

To build a block list, an explicit `*.*` must be used as the first rule, otherwise all tables will be excluded.

```
#### every table will be filtered out
./dumpling -f '!*.Password'

#### only the "Password" table is filtered out, the rest are included.
./dumpling -f '*.*' -f '!*.Password'
```

In a filter list, if a table name matches multiple patterns, the last match decides the outcome. For instance:

```
#### rule 1
employees.*
#### rule 2
```

```
!*.dep*
#### rule 3
*.departments
```

The filtered outcome is as follows:

Table name	Rule 1	Rule 2	Rule 3	Outcome
irrelevant.table				Default (reject)
employees.employees				Rule 1 (accept)
employees.dept_emp				Rule 2 (reject)
employees.departments				Rule 3 (accept)
else.departments				Rule 3 (accept)

Note:

In TiDB tools, the system schemas are always excluded regardless of the table filter settings. The system schemas are:

- `INFORMATION_SCHEMA`
- `PERFORMANCE_SCHEMA`
- `METRICS_SCHEMA`
- `INSPECTION_SCHEMA`
- `mysql`
- `sys`

4.16.9.6 TiDB Lightning CSV Support

TiDB Lightning supports reading CSV (comma-separated values) data source, as well as other delimited format such as TSV (tab-separated values).

4.16.9.6.1 File name

A CSV file representing a whole table must be named as `db_name.table_name.csv`. This will be restored as a table `table_name` inside the database `db_name`.

If a table spans multiple CSV files, they should be named like `db_name.table_name.003.csv`.

The file extension must be `*.csv`, even if the content is not separated by commas.

4.16.9.6.2 Schema

CSV files are schema-less. To import them into TiDB, a table schema must be provided. This could be done either by:

- Providing a file named `db_name.table_name-schema.sql` containing the `CREATE TABLE` DDL statement
- Creating the empty tables directly in TiDB in the first place, and then setting `[mydumper] no-schema = true` in `tidb-lightning.toml`.

4.16.9.6.3 Configuration

The CSV format can be configured in `tidb-lightning.toml` under the `[mydumper.csv]` section. Most settings have a corresponding option in the MySQL `LOAD DATA` statement.

```
[mydumper.csv]
#### Separator between fields, should be an ASCII character.
separator = ','
#### Quoting delimiter, can either be an ASCII character or empty string.
delimiter = '"'
#### Whether the CSV files contain a header.
#### If `header` is true, the first line will be skipped.
header = true
#### Whether the CSV contains any NULL value.
#### If `not-null` is true, all columns from CSV cannot be NULL.
not-null = false
#### When `not-null` is false (that is, CSV can contain NULL),
#### fields equal to this value will be treated as NULL.
null = '\N'
#### Whether to interpret backslash escapes inside fields.
backslash-escape = true
#### If a line ends with a separator, remove it.
trim-last-separator = false
```

`separator`

- Defines the field separator.
- Must be a single ASCII character.
- Common values:
  - `','` for CSV
  - `"\t"` for TSV
- Corresponds to the `FIELDS TERMINATED BY` option in the LOAD DATA statement.

`delimiter`

- Defines the delimiter used for quoting.

- If `delimiter` is empty, all fields are unquoted.
- Common values:
  - `'"'` quote fields with double-quote, same as RFC 4180
  - `''` disable quoting
- Corresponds to the `FIELDS ENCLOSED BY` option in the `LOAD DATA` statement.

`header`

- Whether *all* CSV files contain a header row.
- If `header` is true, the first row will be used as the *column names*. If `header` is false, the first row is not special and treated as an ordinary data row.

`not-null` and `null`

- The `not-null` setting controls whether all fields are non-nullable.
- If `not-null` is false, the string specified by `null` will be transformed to the SQL NULL instead of a concrete value.
- Quoting will not affect whether a field is null.

  For example, with the CSV file:

```
A,B,C
\N,"\N",
```

  In the default settings (`not-null = false; null = '\N'`), the columns `A` and `B` are both converted to NULL after importing to TiDB. The column `C` is simply the empty string `''` but not NULL.

`backslash-escape`

- Whether to interpret backslash escapes inside fields.
- If `backslash-escape` is true, the following sequences are recognized and transformed:

Sequence	Converted to
`\0`	Null character (U+0000)
`\b`	Backspace (U+0008)
`\n`	Line feed (U+000A)
`\r`	Carriage return (U+000D)
`\t`	Tab (U+0009)
`\Z`	Windows EOF (U+001A)

In all other cases (for example, \") the backslash is simply stripped, leaving the next character (") in the field. The character left has no special roles (for example, delimiters) and is just an ordinary character.

- Quoting will not affect whether backslash escapes are interpreted.
- Corresponds to the `FIELDS ESCAPED BY '\'` option in the `LOAD DATA` statement.

`trim-last-separator`

- Treats the field `separator` as a terminator, and removes all trailing separators.

  For example, with the CSV file:

```
A,,B,,
```

- When `trim-last-separator = false`, this is interpreted as a row of 5 fields `('A', '', 'B', '', '')`.
- When `trim-last-separator = true`, this is interpreted as a row of 3 fields `('A', '', 'B')`.

Non-configurable options

TiDB Lightning does not support every option supported by the `LOAD DATA` statement. Some examples:

- The line terminator must only be CR (`\r`), LF (`\n`) or CRLF (`\r\n`), which means `LINES TERMINATED BY` is not customizable.
- There cannot be line prefixes (`LINES STARTING BY`).
- The header cannot be simply skipped (`IGNORE n LINES`), it must be valid column names if present.
- Delimiters and separators can only be a single ASCII character.

4.16.9.6.4 Common configurations

CSV

The default setting is already tuned for CSV following RFC 4180.

```
[mydumper.csv]
separator = ','
delimiter = '"'
header = true
not-null = false
null = '\N'
backslash-escape = true
trim-last-separator = false
```

Example content:

```
ID,Region,Count
1,"East",32
2,"South",\N
3,"West",10
4,"North",39
```

TSV

```
[mydumper.csv]
separator = "\t"
delimiter = ''
header = true
not-null = false
null = 'NULL'
backslash-escape = false
trim-last-separator = false
```

Example content:

```
ID    Region    Count
1     East      32
2     South     NULL
3     West      10
4     North     39
```

TPC-H DBGEN

```
[mydumper.csv]
separator = '|'
delimiter = ''
header = false
not-null = true
backslash-escape = false
trim-last-separator = true
```

Example content:

```
1|East|32|
2|South|0|
3|West|10|
4|North|39|
```

4.16.9.7 TiDB Lightning TiDB-backend

TiDB Lightning supports two backends: Importer and TiDB. It determines how `tidb-lightning` delivers data into the target cluster.

The Importer-backend (default) requires `tidb-lightning` to first encode the SQL or CSV data into KV pairs, and relies on the external `tikv-importer` program to sort these KV pairs and ingest directly into the TiKV nodes.

The TiDB-backend requires `tidb-lightning` to encode these data into SQL `INSERT` statements, and has these statements executed directly on the TiDB node.

Back end	Importer	TiDB
Speed	Fast (~300 GB/hr)	Slow (~50 GB/hr)
Resource usage	High	Low
ACID respected while importing	No	Yes
Target tables	Must be empty	Can be populated

4.16.9.7.1 Deployment for TiDB-backend

When using the TiDB-backend, you no longer need `tikv-importer`. Compared with the standard deployment procedure, the TiDB-backend deployment has the following two differences:

- Steps involving `tikv-importer` can all be skipped.
- The configuration must be changed to indicate the TiDB-backend is used.

Hardware requirements

The speed of TiDB Lightning using TiDB-backend is limited by the SQL processing speed of TiDB. Therefore, even a lower-end machine may max out the possible performance. The recommended hardware configuration is:

- 16 logical cores CPU
- An SSD large enough to store the entire data source, preferring higher read speed
- 1 Gigabit network card

Ansible deployment

1. The `[importer_server]` section in `inventory.ini` can be left blank.

```
...

[importer_server]
# keep empty

[lightning_server]
```

```
192.168.20.10

...
```

2. The `tikv_importer_port` setting in `group_vars/all.yml` is ignored, and the file `group_vars/importer_server.yml` does not need to be changed. But you need to edit `conf/tidb-lightning.yml` and change the `backend` setting to `tidb`.

```
...
tikv_importer:
    backend: "tidb" # <-- change this
...
```

3. Bootstrap and deploy the cluster as usual.
4. Mount the data source for TiDB Lightning as usual.
5. Start `tidb-lightning` as usual.

Manual deployment

You do not need to download and configure `tikv-importer`. You can download TiDB Lightning from here.

Before running `tidb-lightning`, add the following lines into the configuration file:

```
[tikv-importer]
backend = "tidb"
```

or supplying the `--backend tidb` arguments when executing `tidb-lightning`.

4.16.9.7.2 Conflict resolution

The TiDB-backend supports importing to an already-populated table. However, the new data might cause a unique key conflict with the old data. You can control how to resolve the conflict by using this task configuration.

```
[tikv-importer]
backend = "tidb"
on-duplicate = "replace" # or "error" or "ignore"
```

Setting	Behavior on conflict	Equivalent SQL statement
replace	New entries replace old ones	`REPLACE INTO ...`
ignore	Keep old entries and ignore new ones	`INSERT IGNORE INTO ...`
error	Abort import	`INSERT INTO ...`

4.16.9.7.3 Migrating from Loader to TiDB Lightning TiDB-backend

TiDB Lightning using the TiDB-backend can completely replace functions of Loader. The following list shows how to translate Loader configurations into TiDB Lightning configurations.

Loader

TiDB Lightning

```

#### logging
log-level = "info"
log-file = "loader.log"

#### Prometheus
status-addr = ":8272"

#### concurrency
pool-size = 16
```

```
[lightning]
#### logging
level = "info"
file = "tidb-lightning.log"

#### Prometheus
pprof-port = 8289

#### concurrency (better left as default)
#region-concurrency = 16
```

```

#### checkpoint database

checkpoint-schema = "tidb_loader"
```

```
[checkpoint]
#### checkpoint storage
enable = true
schema = "tidb_lightning_checkpoint"
#### by default the checkpoint is stored in
#### a local file, which is more efficient.
#### but you could still choose to store the
#### checkpoints in the target database with
#### this setting:
#driver = "mysql"
```

```
[tikv-importer]
#### use the TiDB-backend
backend = "tidb"
```

```

#### data source directory
dir = "/data/export/"
```

```
[mydumper]
#### data source directory
data-source-dir = "/data/export"
```

```
[db]
#### TiDB connection parameters
host = "127.0.0.1"
port = 4000

user = "root"
password = ""

#sql-mode = ""
```

```
[tidb]
#### TiDB connection parameters
host = "127.0.0.1"
port = 4000
status-port = 10080 # <- this is required
user = "root"
password = ""

#sql-mode = ""
```

```
#### [[route-rules]]
#### Table routes
#### schema-pattern = "shard_db_*"
#### table-pattern = "shard_table_*"
#### target-schema = "shard_db"
#### target-table = "shard_table"
```

```
#### [[routes]]
#### schema-pattern = "shard_db_*"
#### table-pattern = "shard_table_*"
```

```
#### target-schema = "shard_db"
#### target-table = "shard_table"
```

4.16.9.8 TiDB Lightning Web Interface

TiDB Lightning provides a webpage for viewing the import progress and performing some simple task management. This is called the *server mode*.

To enable server mode, either start `tidb-lightning` with the `--server-mode` flag

```
./tidb-lightning --server-mode --status-addr :8289
```

or set the `lightning.server-mode` setting in the configuration file.

```
[lightning]
server-mode = true
status-addr = ':8289'
```

After TiDB Lightning is launched, visit `http://127.0.0.1:8289` to control the program (the actual URL depends on the `status-addr` setting).

In server mode, TiDB Lightning does not start running immediately. Rather, users submit (multiple) *tasks* via the web interface to import data.

4.16.9.8.1 Front page

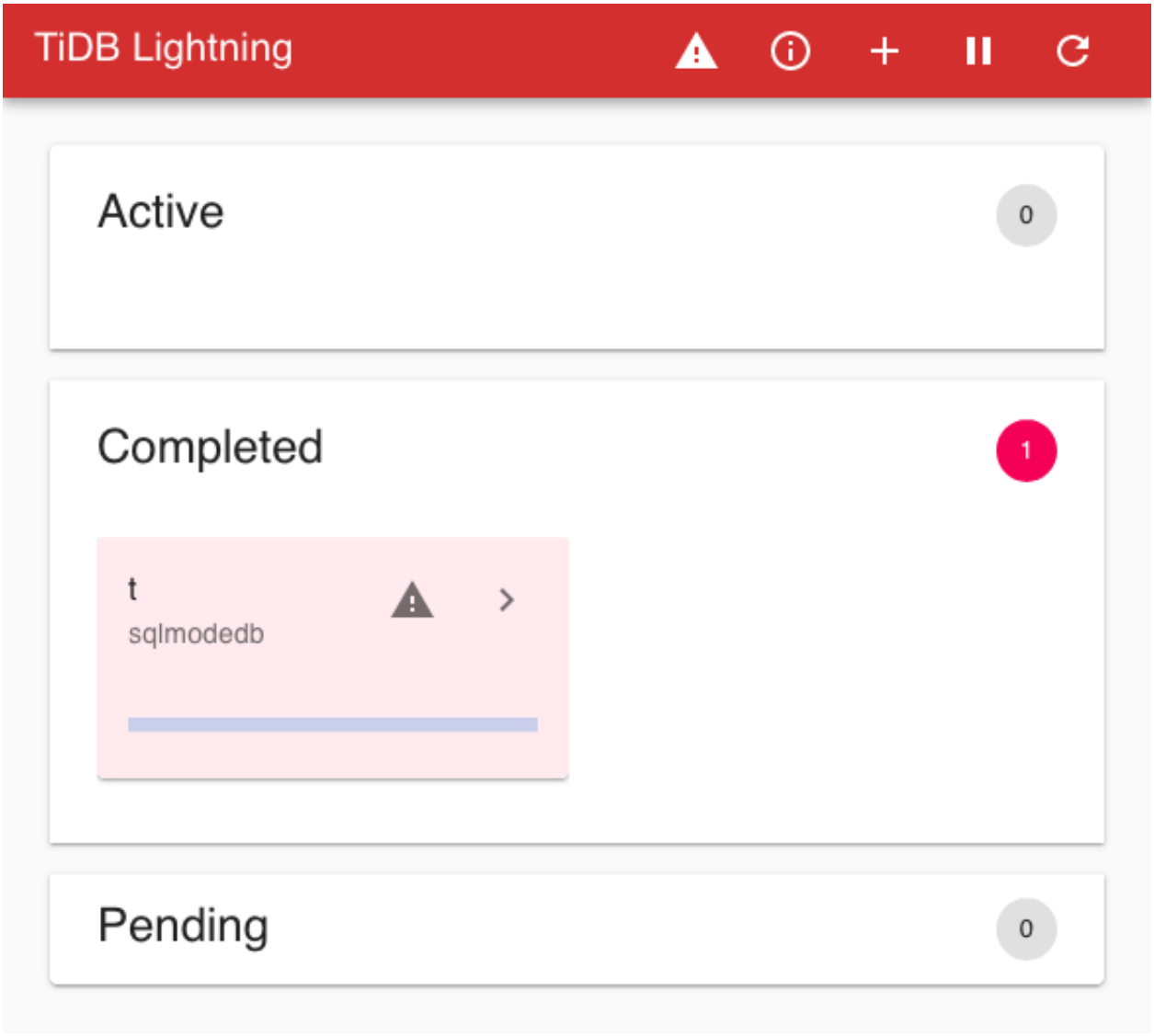

Figure 337: Front page of the web interface

Functions of the title bar, from left to right:

Icon	Function
"TiDB Lightning"	Click to go back to the front page

Icon	Function
	Display any error message from *previous* task
	List current and queued tasks; a badge may appear here to indicate number of queued tasks
+	Submit a task
/	Pause/resume current execution
	Configure auto-refresh of the web page

Three panels below the title bar show all tables in different states:

- Active: these tables are currently being imported
- Completed: these tables have been imported successfully or failed
- Pending: these tables are not yet processed

Each panel contains cards describing the status of the table.

4.16.9.8.2 Submit task

Click the + button on the title bar to submit a task.

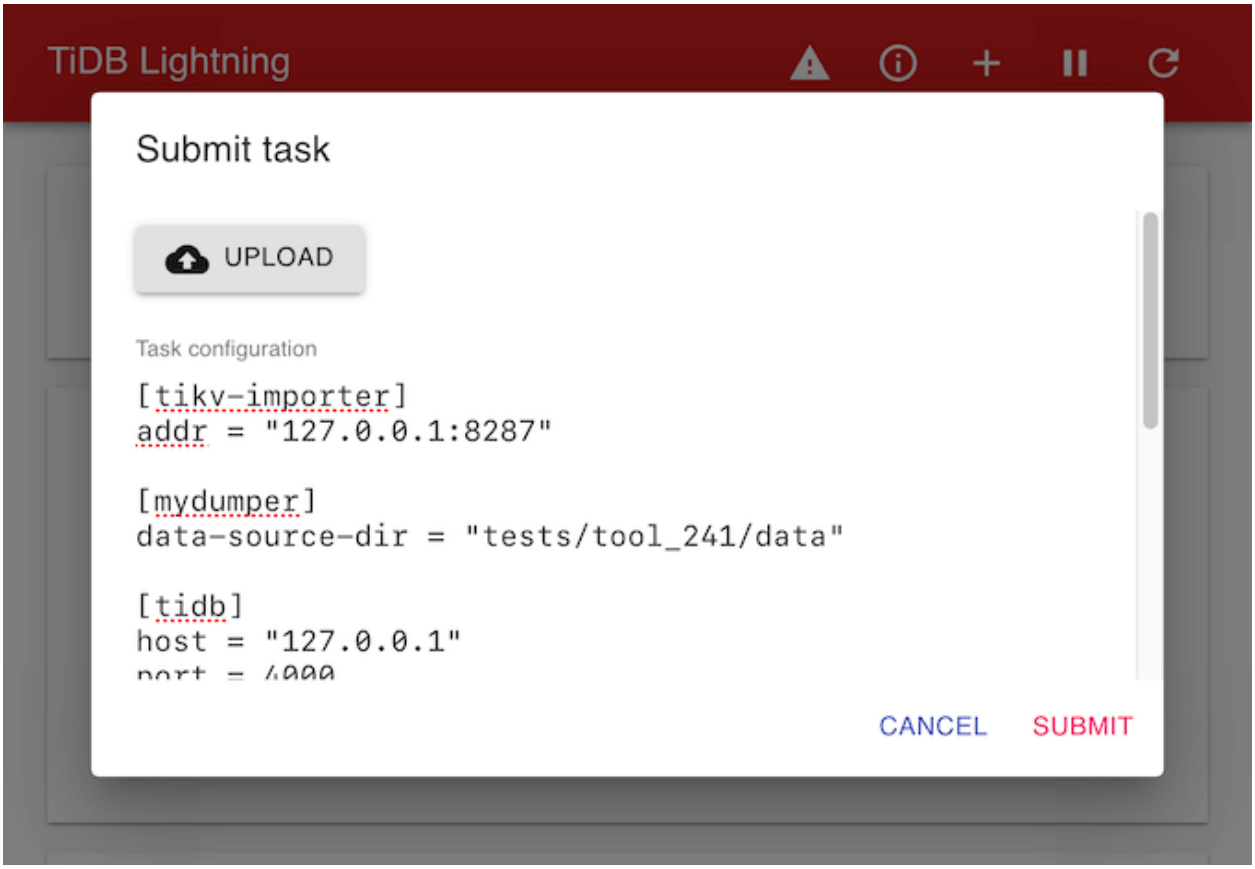

Figure 338: Submit task dialog

Tasks are TOML files described as task configurations. One could also open a local TOML file by clicking **UPLOAD**.

Click **SUBMIT** to run the task. If a task is already running, the new task will be queued and executed after the current task succeeds.

4.16.9.8.3 Table progress

Click the > button of a table card on the front page to view the detailed progress of a table.

Figure 339: Table progress

The page shows the import progress of every engine and data files associated with the table.

Click **TiDB Lightning** on the title bar to go back to the front page.

4.16.9.8.4 Task management

Click the button on the title bar to manage the current and queued tasks.

Figure 340: Task management page

Each task is labeled by the time it was submitted. Clicking the task would show the configuration formatted as JSON.

Manage tasks by clicking the button next to a task. You can stop a task immediately, or reorder queued tasks.

4.16.9.9 TiDB Lightning Monitoring

Both `tidb-lightning` and `tikv-importer` supports metrics collection via Prometheus. This document introduces the monitor configuration and monitoring metrics of TiDB Lightning.

4.16.9.9.1 Monitor configuration

- If TiDB Lightning is installed using TiDB Ansible, simply add the servers to the `[ ↪ monitored_servers]` section in the `inventory.ini` file. Then the Prometheus server can collect their metrics.
- If TiDB Lightning is manually installed, follow the instructions below.

`tikv-importer`

`tikv-importer` v2.1 uses Pushgateway to deliver metrics. Configure `tikv-importer.toml` to recognize the Pushgateway with the following settings:

```
[metric]

#### The Prometheus client push job name.
job = "tikv-importer"

#### The Prometheus client push interval.
interval = "15s"

#### The Prometheus Pushgateway address.
address = ""
```

`tidb-lightning`

The metrics of `tidb-lightning` can be gathered directly by Prometheus as long as it is discovered. You can set the metrics port in `tidb-lightning.toml`:

```
[lightning]
#### HTTP port for debugging and Prometheus metrics pulling (0 to disable)
pprof-port = 8289

...
```

You need to configure Prometheus to make it discover the `tidb-lightning` server. For instance, you can directly add the server address to the `scrape_configs` section:

```
...
scrape_configs:
  - job_name: 'tidb-lightning'
    static_configs:
      - targets: ['192.168.20.10:8289']
```

4.16.9.9.2 Grafana dashboard

Grafana is a web interface to visualize Prometheus metrics as dashboards.

If TiDB Lightning is installed using TiDB Ansible, its dashboard is already installed. Otherwise, the dashboard JSON can be imported from https://raw.githubusercontent.com/pingcap/tidb-ansible/master/scripts/lightning.json.

Row 1: Speed

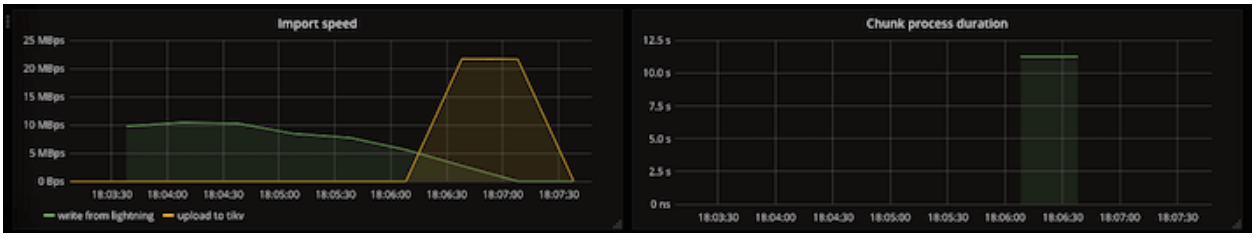

Figure 341: Panels in first row

Panel	Series	Description
Import speed	write from lightning	Speed of sending KVs from TiDB Lightning to TiKV Importer, which depends on each table's complexity
Import speed	upload to tikv	Total upload speed from TiKV Importer to all TiKV replicas

Panel	Series	Description
Chunk process duration		Average time needed to completely encode one single data file

Sometimes the import speed will drop to zero allowing other parts to catch up. This is normal.

Row 2: Progress

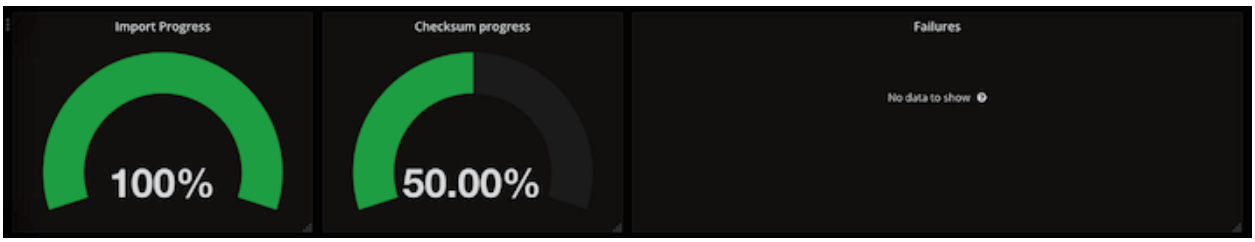

Figure 342: Panels in second row

Panel	Description
Import progress	Percentage of data files encoded so far
Checksum progress	Percentage of tables are verified to be imported successfully
Failures	Number of failed tables and their point of failure, normally empty

Row 3: Resource

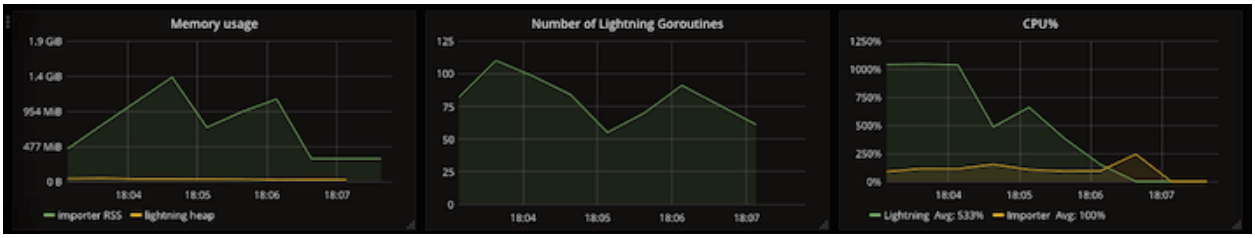

Figure 343: Panels in third row

Panel	Description
Memory usage	Amount of memory occupied by each service
Number of Lightning Goroutines	Number of running goroutines used by TiDB Lightning
CPU%	Number of logical CPU cores utilized by each service

Row 4: Quota

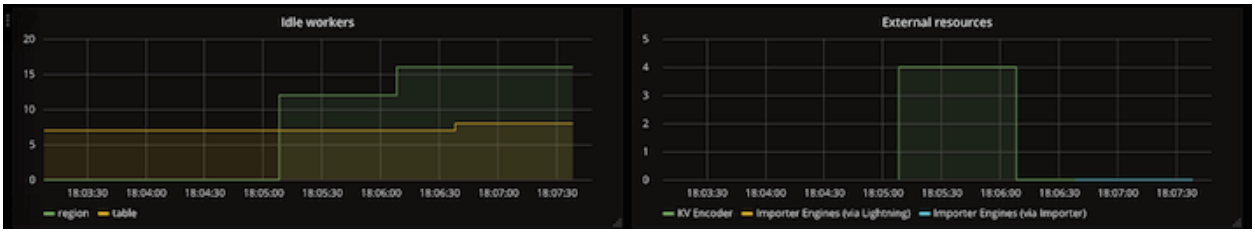

Figure 344: Panels in fourth row

Panel	Series	Description
Idle workers	io	Number of unused `io-concurrency`, normally close to configured value (default 5), and close to 0 means the disk is too slow

Panel	Series	Description
Idle workers	closed-engine	Number of engines which is closed but not yet cleaned up, normally close to index + table-concurrency (default 8), and close to 0 means TiDB Lightning is faster than TiKV Importer, which will cause TiDB Lightning to stall

Panel	Series	Description
Idle workers	table	Number of unused `table-concurrency`, normally 0 until the end of process
Idle workers	index	Number of unused `index-concurrency`, normally 0 until the end of process
Idle workers	region	Number of unused `region-concurrency`, normally 0 until the end of process

Panel	Series	Description
External resources	KV Encoder	Counts active KV encoders, normally the same as `region` ↪ `-` ↪ `concurrency` ↪ until the end of process
External resources	Importer Engines	Counts opened engine files, should never exceed the `max-` ↪ `open` ↪ `-` ↪ `engines` ↪ setting

Row 5: Read speed

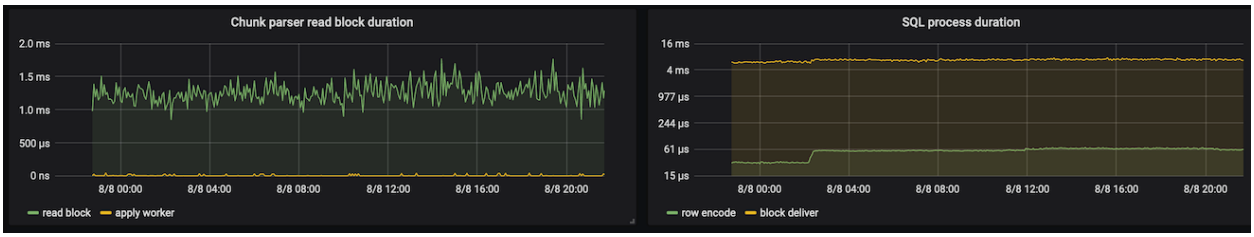

Figure 345: Panels in fifth row

Panel	Series	Description
Chunk parser read block duration	read block	Time taken to read one block of bytes to prepare for parsing
Chunk parser read block duration	apply worker	Time elapsed to wait for an idle io-concurrency
SQL process duration	row encode	Time taken to parse and encode a single row
SQL process duration	block deliver	Time taken to send a block of KV pairs to TiKV Importer

If any of the duration is too high, it indicates that the disk used by TiDB Lightning is too slow or busy with I/O.

Row 6: Storage

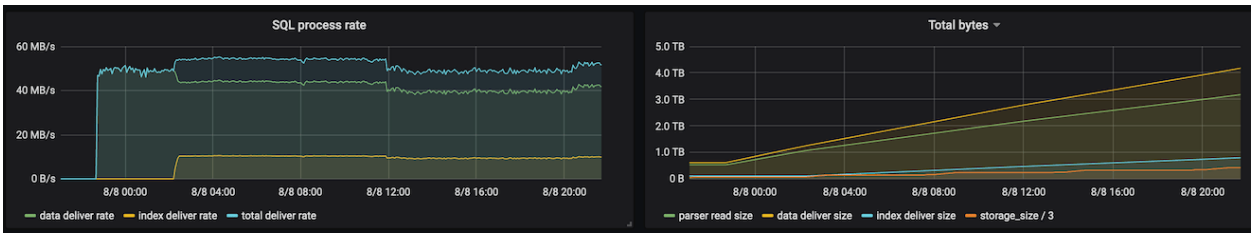

Figure 346: Panels in sixth row

Panel	Series	Description
SQL process rate	data deliver rate	Speed of delivery of data KV pairs to TiKV Importer
SQL process rate	index deliver rate	Speed of delivery of index KV pairs to TiKV Importer
SQL process rate	total deliver rate	The sum of two rates above

Panel	Series	Description
Total bytes	parser read size	Number of bytes being read by TiDB Lightning
Total bytes	data deliver size	Number of bytes of data KV pairs already delivered to TiKV Importer
Total bytes	index deliver size	Number of bytes of index KV pairs already delivered to TiKV Importer

Panel	Series	Description
Total bytes	storage_size / 3	Total size occupied by the TiKV cluster, divided by 3 (the default number of replicas)

Row 7: Import speed

Figure 347: Panels in seventh row

Panel	Series	Description
Delivery duration	Range delivery	Time taken to upload a range of KV pairs to the TiKV cluster

Panel	Series	Description
Delivery duration	SST delivery	Time taken to upload an SST file to the TiKV cluster
SST process duration	Split SST	Time taken to split the stream of KV pairs into SST files
SST process duration	SST upload	Time taken to upload an SST file
SST process duration	SST ingest	Time taken to ingest an uploaded SST file
SST process duration	SST size	File size of an SST file

4.16.9.9.3 Monitoring metrics

This section explains the monitoring metrics of `tikv-importer` and `tidb-lightning`, if you need to monitor other metrics not covered by the default Grafana dashboard.

`tikv-importer`

Metrics provided by `tikv-importer` are listed under the namespace `tikv_import_*`.

- **`tikv_import_rpc_duration`** (Histogram)

  Bucketed histogram for the duration of an RPC action. Labels:

  - **request**: what kind of RPC is executed
    * `switch_mode` — switched a TiKV node to import/normal mode
    * `open_engine` — opened an engine file
    * `write_engine` — received data and written into an engine
    * `close_engine` — closed an engine file
    * `import_engine` — imported an engine file into the TiKV cluster
    * `cleanup_engine` — deleted an engine file
    * `compact_cluster` — explicitly compacted the TiKV cluster
    * `upload` — uploaded an SST file
    * `ingest` — ingested an SST file
    * `compact` — explicitly compacted a TiKV node
  - **result**: the execution result of the RPC
    * `ok`
    * `error`

- **`tikv_import_write_chunk_bytes`** (Histogram)

  Bucketed histogram for the uncompressed size of a block of KV pairs received from Lightning.

- **`tikv_import_write_chunk_duration`** (Histogram)

  Bucketed histogram for the time needed to receive a block of KV pairs from Lightning.

- **`tikv_import_upload_chunk_bytes`** (Histogram)

  Bucketed histogram for the compressed size of a chunk of SST file uploaded to TiKV.

- **`tikv_import_upload_chunk_duration`** (Histogram)

  Bucketed histogram for the time needed to upload a chunk of SST file to TiKV.

- **`tikv_import_range_delivery_duration`** (Histogram)

  Bucketed histogram for the time needed to deliver a range of KV pairs into a `dispatch-job`.

- **`tikv_import_split_sst_duration`** (Histogram)

  Bucketed histogram for the time needed to split off a range from the engine file into a single SST file.

- **`tikv_import_sst_delivery_duration`** (Histogram)

  Bucketed histogram for the time needed to deliver an SST file from a `dispatch-job` to an `ImportSSTJob`.

- **tikv_import_sst_recv_duration** (Histogram)

  Bucketed histogram for the time needed to receive an SST file from a `dispatch-job` in an `ImportSSTJob`.

- **tikv_import_sst_upload_duration** (Histogram)

  Bucketed histogram for the time needed to upload an SST file from an `ImportSSTJob` to a TiKV node.

- **tikv_import_sst_chunk_bytes** (Histogram)

  Bucketed histogram for the compressed size of the SST file uploaded to a TiKV node.

- **tikv_import_sst_ingest_duration** (Histogram)

  Bucketed histogram for the time needed to ingest an SST file into TiKV.

- **tikv_import_each_phase** (Gauge)

  Indicates the running phase. Possible values are 1, meaning running inside the phase, and 0, meaning outside the phase. Labels:

  - **phase**: `prepare/import`

- **tikv_import_wait_store_available_count** (Counter)

  Counts the number of times a TiKV node is found to have insufficient space when uploading SST files. Labels:

  - **store_id**: The TiKV store ID.

`tidb-lightning`

Metrics provided by `tidb-lightning` are listed under the namespace `lightning_*`.

- **lightning_importer_engine** (Counter)

  Counts open and closed engine files. Labels:

  - **type**:
    * `open`
    * `closed`

- **lightning_idle_workers** (Gauge)

  Counts idle workers. Labels:

  - **name**:
    * `table` — the remainder of `table-concurrency`, normally 0 until the end of the process

    * `index` — the remainder of `index-concurrency`, normally 0 until the end of the process
    * `region` — the remainder of `region-concurrency`, normally 0 until the end of the process
    * `io` — the remainder of `io-concurrency`, normally close to configured value (default 5), and close to 0 means the disk is too slow
    * `closed-engine` — number of engines which have been closed but not yet cleaned up, normally close to index + table-concurrency (default 8). A value close to 0 means TiDB Lightning is faster than TiKV Importer, which might cause TiDB Lightning to stall

- **`lightning_kv_encoder`** (Counter)

  Counts open and closed KV encoders. KV encoders are in-memory TiDB instances that convert SQL `INSERT` statements into KV pairs. The net values need to be bounded in a healthy situation. Labels:

  - **type**:
    * `open`
    * `closed`

- **`lightning_tables`** (Counter)

  Counts processed tables and their statuses. Labels:

  - **state**: the status of the table, indicating which phase should be completed
    * `pending` — not yet processed
    * `written` — all data encoded and sent
    * `closed` — all corresponding engine files closed
    * `imported` — all engine files have been imported into the target cluster
    * `altered_auto_inc` — AUTO_INCREMENT ID altered
    * `checksum` — checksum performed
    * `analyzed` — statistics analysis performed
    * `completed` — the table has been fully imported and verified
  - **result**: the result of the current phase
    * `success` — the phase completed successfully
    * `failure` — the phase failed (did not complete)

- **`lightning_engines`** (Counter)

  Counts number of engine files processed and their status. Labels:

  - **state**: the status of the engine, indicating which phase should be completed
    * `pending` — not yet processed
    * `written` — all data encoded and sent
    * `closed` — engine file closed

    * `imported` — the engine file has been imported into the target cluster
    * `completed` — the engine has been fully imported
  - **result**: the result of the current phase
    * `success` — the phase completed successfully
    * `failure` — the phase failed (did not complete)

- **`lightning_chunks`** (Counter)

  Counts number of chunks processed and their status. Labels:

  - **state**: a chunk's status, indicating which phase the chunk is in
    * `estimated` — (not a state) this value gives total number of chunks in current task
    * `pending` — loaded but not yet processed
    * `running` — data are being encoded and sent
    * `finished` — the entire chunk has been processed
    * `failed` — errors happened during processing

- **`lightning_import_seconds`** (Histogram)

  Bucketed histogram for the time needed to import a table.

- **`lightning_row_read_bytes`** (Histogram)

  Bucketed histogram for the size of a single SQL row.

- **`lightning_row_encode_seconds`** (Histogram)

  Bucketed histogram for the time needed to encode a single SQL row into KV pairs.

- **`lightning_row_kv_deliver_seconds`** (Histogram)

  Bucketed histogram for the time needed to deliver a set of KV pairs corresponding to one single SQL row.

- **`lightning_block_deliver_seconds`** (Histogram)

  Bucketed histogram for the time needed to deliver a block of KV pairs to Importer.

- **`lightning_block_deliver_bytes`** (Histogram)

  Bucketed histogram for the uncompressed size of a block of KV pairs delivered to Importer.

- **`lightning_chunk_parser_read_block_seconds`** (Histogram)

  Bucketed histogram for the time needed by the data file parser to read a block.

- **`lightning_checksum_seconds`** (Histogram)

  Bucketed histogram for the time needed to compute the checksum of a table.

- **`lightning_apply_worker_seconds`** (Histogram)

  Bucketed histogram for the time needed to acquire an idle worker (see also the `lightning_idle_workers` gauge). Labels:

  - **name**:
    * `table`
    * `index`
    * `region`
    * `io`
    * `closed-engine`

4.16.9.10 TiDB Lightning FAQ

4.16.9.10.1 What is the minimum TiDB/TiKV/PD cluster version supported by Lightning?

The version of TiDB Lightning should be the same as the cluster. The earliest available version is 2.0.9, but we recommend using the latest stable version (3.0).

4.16.9.10.2 Does TiDB Lightning support importing multiple schemas (databases)?

Yes.

4.16.9.10.3 What is the privilege requirements for the target database?

TiDB Lightning requires the following privileges:

- SELECT
- UPDATE
- ALTER
- CREATE
- DROP

If the TiDB-backend is chosen, or the target database is used to store checkpoints, it additionally requires these privileges:

- INSERT
- DELETE

The Importer-backend does not require these two privileges because data is ingested into TiKV directly, which bypasses the entire TiDB privilege system. This is secure as long as the ports of TiKV, TiKV Importer and TiDB Lightning are not reachable outside the cluster.

If the `checksum` configuration of TiDB Lightning is set to `true`, then the admin user privileges in the downstream TiDB need to be granted to TiDB Lightning.

4.16.9.10.4 TiDB Lightning encountered an error when importing one table. Will it affect other tables? Will the process be terminated?

If only one table has an error encountered, the rest will still be processed normally.

4.16.9.10.5 How to properly restart TiDB Lightning?

Depending on the status of `tikv-importer`, the basic sequence of restarting TiDB Lightning is like this:

If `tikv-importer` is still running:

1. Stop `tidb-lightning`.
2. Perform the intended modifications, such as fixing the source data, changing settings, replacing hardware etc.
3. If the modification previously has changed any table, remove the corresponding checkpoint too.
4. Start `tidb-lightning`.

If `tikv-importer` needs to be restarted:

1. Stop `tidb-lightning`.
2. Stop `tikv-importer`.
3. Perform the intended modifications, such as fixing the source data, changing settings, replacing hardware etc.
4. Start `tikv-importer`.
5. Start `tidb-lightning` *and wait until the program fails with CHECKSUM error, if any.*
    - Restarting `tikv-importer` would destroy all engine files still being written, but `tidb-lightning` did not know about it. As of v3.0 the simplest way is to let `tidb-lightning` go on and retry.
6. Destroy the failed tables and checkpoints
7. Start `tidb-lightning` again.

4.16.9.10.6 How to ensure the integrity of the imported data?

TiDB Lightning by default performs checksum on the local data source and the imported tables. If there is checksum mismatch, the process would be aborted. These checksum information can be read from the log.

You could also execute the `ADMIN CHECKSUM TABLE` SQL command on the target table to recompute the checksum of the imported data.

```
ADMIN CHECKSUM TABLE `schema`.`table`;
```

```
+---------+------------+--------------------+-----------+-------------+
| Db_name | Table_name | Checksum_crc64_xor | Total_kvs | Total_bytes |
+---------+------------+--------------------+-----------+-------------+
| schema | table     | 5505282386844578743 |     3 |         96 |
+---------+------------+--------------------+-----------+-------------+
1 row in set (0.01 sec)
```

4.16.9.10.7 What kind of data source format is supported by Lightning?

TiDB Lightning only supports the SQL dump generated by Mydumper or CSV files stored in the local file system.

4.16.9.10.8 Could TiDB Lightning skip creating schema and tables?

Yes. If you have already created the tables in the target database, you could set `no-schema = true` in the `[mydumper]` section in `tidb-lightning.toml`. This makes TiDB Lightning skip the `CREATE TABLE` invocations and fetch the metadata directly from the target database. TiDB Lightning will exit with error if a table is actually missing.

4.16.9.10.9 Can the Strict SQL Mode be disabled to allow importing invalid data?

Yes. By default, the `sql_mode` used by TiDB Lightning is `"STRICT_TRANS_TABLES, NO_ENGINE_SUBSTITUTION"`, which disallows invalid data such as the date `1970-00-00`. The mode can be changed by modifying the `sql-mode` setting in the `[tidb]` section in `tidb-lightning.toml`.

```
...
[tidb]
sql-mode = ""
...
```

4.16.9.10.10 Can one `tikv-importer` serve multiple `tidb-lightning` instances?

Yes, as long as every `tidb-lightning` instance operates on different tables.

4.16.9.10.11 How to stop the `tikv-importer` process?

To stop the `tikv-importer` process, you can choose the corresponding operation according to your deployment method.

- For deployment using TiDB Ansible: run `scripts/stop_importer.sh` on the Importer server.

- For manual deployment: if `tikv-importer` is running in foreground, press Ctrl+C to exit. Otherwise, obtain the process ID using the `ps aux | grep tikv-importer` command and then terminate the process using the `kill «pid»` command.

4.16.9.10.12 How to stop the `tidb-lightning` process?

To stop the `tidb-lightning` process, you can choose the corresponding operation according to your deployment method.

- For deployment using TiDB Ansible: run `scripts/stop_lightning.sh` on the Lightning server.
- For manual deployment: if `tidb-lightning` is running in foreground, press Ctrl+C to exit. Otherwise, obtain the process ID using the `ps aux | grep tidb-lighting` command and then terminate the process using the `kill -2 «pid»` command.

4.16.9.10.13 Why the `tidb-lightning` process suddenly quits while running in background?

It is potentially caused by starting `tidb-lightning` incorrectly, which causes the system to send a SIGHUP signal to stop the `tidb-lightning` process. In this situation, `tidb-lightning.log` usually outputs the following log:

```
[2018/08/10 07:29:08.310 +08:00] [INFO] [main.go:41] ["got signal to exit"]
    ↪ [signal=hangup]
```

It is not recommended to directly use `nohup` in the command line to start `tidb-lightning`. You can start `tidb-lightning` by executing a script.

4.16.9.10.14 Why my TiDB cluster is using lots of CPU resources and running very slowly after using TiDB Lightning?

If `tidb-lightning` abnormally exited, the cluster might be stuck in the "import mode", which is not suitable for production. You can force the cluster back to "normal mode" using the following command:

```
tidb-lightning-ctl --switch-mode=normal
```

4.16.9.10.15 Can TiDB Lightning be used with 1-Gigabit network card?

The TiDB Lightning toolset is best used with a 10-Gigabit network card. 1-Gigabit network cards are *not recommended*, especially for `tikv-importer`.

1-Gigabit network cards can only provide a total bandwidth of 120 MB/s, which has to be shared among all target TiKV stores. TiDB Lightning can easily saturate all bandwidth of the 1-Gigabit network and bring down the cluster because PD is unable to be contacted anymore. To avoid this, set an *upload speed limit* in Importer's configuration:

```
[import]
#### Restricts the total upload speed to TiKV to 100 MB/s or less
upload-speed-limit = "100MB"
```

4.16.9.10.16 Why TiDB Lightning requires so much free space in the target TiKV cluster?

With the default settings of 3 replicas, the space requirement of the target TiKV cluster is 6 times the size of data source. The extra multiple of "2" is a conservative estimation because the following factors are not reflected in the data source:

- The space occupied by indices
- Space amplification in RocksDB

4.16.9.10.17 Can TiKV Importer be restarted while TiDB Lightning is running?

No. Importer stores some information of engines in memory. If `tikv-importer` is restarted, `tidb-lightning` will be stopped due to lost connection. At this point, you need to destroy the failed checkpoints as those Importer-specific information is lost. You can restart Lightning afterwards.

See also How to properly restart TiDB Lightning? for the correct sequence.

4.16.9.10.18 How to completely destroy all intermediate data associated with TiDB Lightning?

1. Delete the checkpoint file.

   ```
   tidb-lightning-ctl --config conf/tidb-lightning.toml --checkpoint-
       ↪ remove=all
   ```

   If, for some reason, you cannot run this command, try manually deleting the file `/tmp` ↪ `/tidb_lightning_checkpoint.pb`.

2. If you are using Local-backend, delete the `sorted-kv-dir` directory in the configuration. If you are using Importer-backend, delete the entire `import` directory on the machine hosting `tikv-importer`.

3. Delete all tables and databases created on the TiDB cluster, if needed.

4.16.9.10.19 Why does TiDB Lightning report the `could not find first pair, this shouldn't happen` error?

This error occurs possibly because the number of files opened by TiDB Lightning exceeds the system limit when TiDB Lightning reads the sorted local files. In the Linux system, you can use the `ulimit -n` command to confirm whether the value of this system limit is too small. It is recommended that you adjust this value to `1000000` (`ulimit -n 1000000`) during TiDB Lightning import.

4.16.9.10.20 Import speed is too slow

Normally it takes TiDB Lightning 2 minutes per thread to import a 256 MB data file. If the speed is much slower than this, there is an error. You can check the time taken for each data file from the log mentioning `restore chunk ⋯ takes`. This can also be observed from metrics on Grafana.

There are several reasons why TiDB Lightning becomes slow:

Cause 1: `region-concurrency` is set too high, which causes thread contention and reduces performance.

1. The setting can be found from the start of the log by searching `region-concurrency`.
2. If TiDB Lightning shares the same machine with other services (for example, TiKV Importer), `region-concurrency` must be **manually** set to 75% of the total number of CPU cores.
3. If there is a quota on CPU (for example, limited by Kubernetes settings), TiDB Lightning may not be able to read this out. In this case, `region-concurrency` must also be **manually** reduced.

Cause 2: The table schema is too complex.

Every additional index introduces a new KV pair for each row. If there are N indices, the actual size to be imported would be approximately (N+1) times the size of the Dumpling output. If the indices are negligible, you may first remove them from the schema, and add them back using `CREATE INDEX` after the import is complete.

Cause 3: Each file is too large.

TiDB Lightning works the best when the data source is broken down into multiple files of size around 256 MB so that the data can be processed in parallel. If each file is too large, TiDB Lightning might not respond.

If the data source is CSV, and all CSV files have no fields containing newline control characters (U+000A and U+000D), you can turn on "strict format" to let TiDB Lightning automatically split the large files.

```
[mydumper]
strict-format = true
```

Cause 4: TiDB Lightning is too old.

Try the latest version! Maybe there is new speed improvement.

4.16.9.10.21 `checksum failed: checksum mismatched remote vs local`

Cause: The checksum of a table in the local data source and the remote imported database differ. This error has several deeper reasons:

1. The table might already have data before. These old data can affect the final checksum.
2. If the remote checksum is 0, which means nothing is imported, it is possible that the cluster is too hot and fails to take in any data.
3. If the data is mechanically generated, ensure it respects the constrains of the table:
    - `AUTO_INCREMENT` columns need to be positive, and do not contain the value "0".
    - The UNIQUE and PRIMARY KEYs must have no duplicated entries.
4. If TiDB Lightning has failed before and was not properly restarted, a checksum mismatch may happen due to data being out-of-sync.

Solutions:

1. Delete the corrupted data using `tidb-lightning-ctl`, and restart TiDB Lightning to import the affected tables again.

```
tidb-lightning-ctl --config conf/tidb-lightning.toml --checkpoint-error
    ↪ -destroy=all
```

2. Consider using an external database to store the checkpoints (change `[checkpoint]` ↪ `dsn`) to reduce the target database's load.
3. If TiDB Lightning was improperly restarted, see also the "How to properly restart TiDB Lightning" section in the FAQ.

4.16.9.10.22 `Checkpoint for … has invalid status:` (error code)

Cause: Checkpoint is enabled, and TiDB Lightning or TiKV Importer has previously abnormally exited. To prevent accidental data corruption, Lightning will not start until the error is addressed.

The error code is an integer smaller than 25, with possible values of 0, 3, 6, 9, 12, 14, 15, 17, 18, 20, and 21. The integer indicates the step where the unexpected exit occurs in the import process. The larger the integer is, the later step the exit occurs at.

Solutions:

If the error was caused by invalid data source, delete the imported data using `tidb-` ↪ `lightning-ctl` and start Lightning again.

```
tidb-lightning-ctl --config conf/tidb-lightning.toml --checkpoint-error-
    ↪ destroy=all
```

See the Checkpoints control section for other options.

4.16.9.10.23 `ResourceTemporarilyUnavailable("Too many open engines …: …")`

Cause: The number of concurrent engine files exceeds the limit specified by `tikv-importer`. This could be caused by misconfiguration. Additionally, if `tidb-lightning` exited abnormally, an engine file might be left at a dangling open state, which could cause this error as well.

Solutions:

1. Increase the value of `max-open-engines` setting in `tikv-importer.toml`. This value is typically dictated by the available memory. This could be calculated by using:

   Max Memory Usage ≈ `max-open-engines` × `write-buffer-size` × `max-write-buffer-number`

2. Decrease the value of `table-concurrency` + `index-concurrency` so it is less than `max-open-engines`.

3. Restart `tikv-importer` to forcefully remove all engine files (default to `./data.import/`). This also removes all partially imported tables, which requires Lightning to clear the outdated checkpoints.

```
tidb-lightning-ctl --config conf/tidb-lightning.toml --checkpoint-error-destroy=all
```

4.16.9.10.24 `cannot guess encoding for input file, please convert to UTF-8 manually`

Cause: TiDB Lightning only recognizes the UTF-8 and GB-18030 encodings for the table schemas. This error is emitted if the file isn't in any of these encodings. It is also possible that the file has mixed encoding, such as containing a string in UTF-8 and another string in GB-18030, due to historical `ALTER TABLE` executions.

Solutions:

1. Fix the schema so that the file is entirely in either UTF-8 or GB-18030.

2. Manually `CREATE` the affected tables in the target database, and then set `[mydumper] no-schema = true` to skip automatic table creation.

3. Set `[mydumper] character-set = "binary"` to skip the check. Note that this might introduce mojibake into the target database.

4.16.9.10.25 [sql2kv] sql encode error = [types:1292]invalid time format: '{1970 1 1 …}'

Cause: A table contains a column with the `timestamp` type, but the time value itself does not exist. This is either because of DST changes or the time value has exceeded the supported range (Jan 1, 1970 to Jan 19, 2038).

Solutions:

1. Ensure Lightning and the source database are using the same time zone.

   When executing Lightning directly, the time zone can be forced using the `$TZ` environment variable.

   ```
   # Manual deployment, and force Asia/Shanghai.
   TZ='Asia/Shanghai' bin/tidb-lightning -config tidb-lightning.toml
   ```

2. When exporting data using Mydumper, make sure to include the `--skip-tz-utc` flag.

3. Ensure the entire cluster is using the same and latest version of `tzdata` (version 2018i or above).

   On CentOS, run `yum info tzdata` to check the installed version and whether there is an update. Run `yum upgrade tzdata` to upgrade the package.

4.16.9.10.26 [Error 8025: entry too large, the max entry size is 6291456]

Cause: A single row of key-value pairs generated by TiDB Lightning exceeds the limit set by TiDB.

Solution:

Currently, the limitation of TiDB cannot be bypassed. You can only ignore this table to ensure the successful import of other tables.

4.16.9.10.27 restore table test.district failed: unknown columns in header [...]

This error occurs usually because the CSV data file does not contain a header (the first row is not column names but data). Therefore, you need to add the following configuration to the TiDB Lightning configuration file:

```
[mydumper.csv]
header = false
```

4.16.9.11 TiDB Lightning Glossary

This page explains the special terms used in TiDB Lightning's logs, monitoring, configurations, and documentation.

4.16.9.11.1 A

Analyze

An operation to rebuild the statistics information of a TiDB table, which is running the `ANALYZE TABLE` statement.

Because TiDB Lightning imports data without going through TiDB, the statistics information is not automatically updated. Therefore, TiDB Lightning explicitly analyzes every table after importing. This step can be omitted by setting the `post-restore.analyze` configuration to `false`.

`AUTO_INCREMENT_ID`

Every table has an associated `AUTO_INCREMENT_ID` counter to provide the default value of an auto-incrementing column. In TiDB, this counter is additionally used to assign row IDs.

Because TiDB Lightning imports data without going through TiDB, the `AUTO_INCREMENT_ID` ↪ counter is not automatically updated. Therefore, TiDB Lightning explicitly alters `AUTO_INCREMENT_ID` to a valid value. This step is always performed, even if the table has no `AUTO_INCREMENT` columns.

4.16.9.11.2 B

Back end

Back end is the destination where TiDB Lightning sends the parsed result. Also spelled as "backend".

See TiDB Lightning TiDB-backend for details.

4.16.9.11.3 C

Checkpoint

TiDB Lightning continuously saves its progress into a local file or a remote database while importing. This allows it to resume from an intermediate state should it crashes in the process. See the Checkpoints section for details.

Checksum

In TiDB Lightning, the checksum of a table is a set of 3 numbers calculated from the content of each KV pair in that table. These numbers are respectively:

- the number of KV pairs,
- total length of all KV pairs, and
- the bitwise-XOR of CRC-64-ECMA value each pair.

TiDB Lightning validates the imported data by comparing the local and remote checksums of every table. The program would stop if any pair does not match. You can skip this check by setting the `post-restore.checksum` configuration to `false`.

See also the FAQs for how to properly handle checksum mismatch.

Chunk

Equivalent to a single file in the data source.

Compaction

An operation that merges multiple small SST files into one large SST file, and cleans up the deleted entries. TiKV automatically compacts data in background while TiDB Lightning is importing.

> **Note:**
>
> For legacy reasons, you can still configure TiDB Lightning to explicitly trigger a compaction every time a table is imported. However, this is not recommended and the corresponding settings are disabled by default.

See RocksDB's wiki page on Compaction for its technical details.

4.16.9.11.4 D

Data engine

An engine for sorting actual row data.

When a table is very large, its data is placed into multiple data engines to improve task pipelining and save space of TiKV Importer. By default, a new data engine is opened for every 100 GB of SQL data, which can be configured through the `mydumper.batch-size` setting.

TiDB Lightning processes multiple data engines concurrently. This is controlled by the `lightning.table-concurrency` setting.

4.16.9.11.5 E

Engine

In TiKV Importer, an engine is a RocksDB instance for sorting KV pairs.

TiDB Lightning transfers data to TiKV Importer through engines. It first opens an engine, sends KV pairs to it (with no particular order), and finally closes the engine. The engine sorts the received KV pairs after it is closed. These closed engines can then be further uploaded to the TiKV stores for ingestion.

Engines use TiKV Importer's `import-dir` as temporary storage, which are sometimes referred to as "engine files".

See also data engine and index engine.

4.16.9.11.6 F

Filter

A configuration list that specifies which tables to be imported or excluded.

See Table Filter for details.

4.16.9.11.7 I

Import mode

A configuration that optimizes TiKV for writing at the cost of degraded read speed and space usage.

TiDB Lightning automatically switches to and off the import mode while running. However, if TiKV gets stuck in import mode, you can use `tidb-lightning-ctl` to force revert to normal mode.

Index engine

An engine for sorting indices.

Regardless of number of indices, every table is associated with exactly one index engine.

TiDB Lightning processes multiple index engines concurrently. This is controlled by the `lightning.index-concurrency` setting. Since every table has exactly one index engine, this also configures the maximum number of tables to process at the same time.

Ingest

An operation which inserts the entire content of an SST file into the RocksDB (TiKV) store.

Ingestion is a very fast operation compared with inserting KV pairs one by one. This operation is the determinant factor for the performance of TiDB Lightning.

See RocksDB's wiki page on Creating and Ingesting SST files for its technical details.

4.16.9.11.8 K

KV pair

Abbreviation of "key-value pair".

KV encoder

A routine which parses SQL or CSV rows to KV pairs. Multiple KV encoders run in parallel to speed up processing.

4.16.9.11.9 L

Local checksum

The checksum of a table calculated by TiDB Lightning itself before sending the KV pairs to TiKV Importer.

4.16.9.11.10 N

Normal mode

The mode where import mode is disabled.

4.16.9.11.11 P

Post-processing

The period of time after the entire data source is parsed and sent to TiKV Importer. TiDB Lightning is waiting for TiKV Importer to upload and ingest the SST files.

4.16.9.11.12 R

Remote checksum

The checksum of a table calculated by TiDB after it has been imported.

4.16.9.11.13 S

Scattering

An operation that randomly reassigns the leader and the peers of a Region. Scattering ensures that the imported data are distributed evenly among TiKV stores. This reduces stress on PD.

Splitting

An engine is typically very large (around 100 GB), which is not friendly to TiKV if treated as a single region. TiKV Importer splits an engine into multiple small SST files (configurable by TiKV Importer's `import.region-split-size` setting) before uploading.

SST file

SST is the abbreviation of "sorted string table". An SST file is RocksDB's (and thus TiKV's) native storage format of a collection of KV pairs.

TiKV Importer produces SST files from a closed engine. These SST files are uploaded and then ingested into TiKV stores.

4.16.10 sync-diff-inspector

4.16.10.1 sync-diff-inspector User Guide

sync-diff-inspector is a tool used to compare data stored in the databases with the MySQL protocol. For example, it can compare the data in MySQL with that in TiDB, the data in MySQL with that in MySQL, or the data in TiDB with that in TiDB. In addition, you can also use this tool to repair data in the scenario where a small amount of data is inconsistent.

This guide introduces the key features of sync-diff-inspector and describes how to configure and use this tool. You can download it at tidb-community-toolkit-v3.0.15-linux-amd64.

4.16.10.1.1 Key features

- Compare the table schema and data
- Generate the SQL statements used to repair data if the data inconsistency exists
- Support data check for tables with different schema or table names
- Support data check in the sharding scenario
- Support data check for TiDB upstream-downstream clusters

4.16.10.1.2 Usage of sync-diff-inspector

Restrictions

- Online check is not supported for data migration between MySQL and TiDB. Ensure that no data is written into the upstream-downstream checklist, and that data in a certain range is not changed. You can check data in this range by setting `range`.
- `JSON`, `BIT`, `BINARY`, `BLOB` and other types of data are not supported. When you perform a data check, you need to set `ignore-columns` to skip checking these types of data.
- In TiDB and MySQL, `FLOAT`, `DOUBLE` and other floating-point types are implemented differently, so checksum might be calculated differently. If the data checks are inconsistent due to these data types, set `ignore-columns` to skip checking these columns.
- Support checking tables that do not contain the primary key or the unique index. However, if data is inconsistent, the generated SQL statements might not be able to repair the data correctly.

Database privilege

sync-diff-inspector needs to obtain the information of table schema, to query data, and to build the `checkpoint` database to save breakpoint information. The required database privileges are as follows:

- Upstream database
  - `SELECT` (checks data for comparison)
  - `SHOW_DATABASES` (views database name)
  - `RELOAD` (views table schema)
- Downstream database
  - `SELECT` (checks data for comparison)
  - `CREATE` (creates the `checkpoint` database and tables)
  - `DELETE` (deletes information in the `checkpoint` table)
  - `INSERT` (writes data into the `checkpoint` table)
  - `UPDATE` (modifies the `checkpoint` table)
  - `SHOW_DATABASES` (views database name)

- `RELOAD` (views table schema)

Configuration file description

The configuration of sync-diff-inspector consists of three parts:

- `Global config`: General configurations, including log level, chunk size, number of threads to check.
- `Tables config`: Configures the tables for checking. If some tables have a certain mapping relationship between the upstream and downstream databases or have some special requirements, you must configure these tables.
- `Databases config`: Configures the instances of the upstream and downstream databases.

Below is the description of a complete configuration file:

```
#### Diff Configuration.

############################# Global config ##########################

#### The log level. You can set it to "info" or "debug".
log-level = "info"

#### sync-diff-inspector divides the data into multiple chunks based on the
    ↪ primary key,
#### unique key, or the index, and then compares the data of each chunk.
#### Uses "chunk-size" to set the size of a chunk.
chunk-size = 1000

#### The number of goroutines created to check data
check-thread-count = 4

#### The proportion of sampling check. If you set it to 100, all the data is
    ↪  checked.
sample-percent = 100

#### If enabled, the chunk's checksum is calculated and data is compared by
    ↪ checksum.
#### If disabled, data is compared line by line.
use-checksum = true

#### If it is set to true, data is checked only by calculating checksum.
    ↪ Data is not checked after inspection, even if the upstream and
    ↪ downstream checksums are inconsistent.
only-use-checksum = false
```

```
#### Whether to use the checkpoint of the last check. If it is enabled, the
    ↪ inspector only checks the last unverified chunks and chunks that
    ↪ failed the verification.
use-checkpoint = true

#### If it is set to true, data check is ignored.
#### If it is set to false, data is checked.
ignore-data-check = false

#### If it is set to true, the table struct comparison is ignored.
#### If set to false, the table struct is compared.
ignore-struct-check = false

#### The name of the file which saves the SQL statements used to repair data
fix-sql-file = "fix.sql"

############################## Tables config ##########################

#### To compare the data of a large number of tables with different schema
    ↪ names or table names, or check the data of multiple upstream sharded
    ↪ tables and downstream table family, use the table-rule to configure
    ↪ the mapping relationship. You can configure the mapping rule only for
    ↪  the schema or table. Also, you can configure the mapping rules for
    ↪ both the schema and the table.
#[[table-rules]]
    # schema-pattern and table-pattern support the wildcard *?
    # schema-pattern = "test_*"
    # table-pattern = "t_*"
    # target-schema = "test"
    # target-table = "t"

#### Configures the tables of the target database that need to be checked.
[[check-tables]]
    # The name of the schema in the target database
    schema = "test"

    # The list of tables that need to be checked in the target database
    tables = ["test1", "test2", "test3"]

    # Supports using regular expressions to configure tables to be checked.
    # You need to start with '~'. For example, the following configuration
        ↪ checks
    # all the tables with the prefix 'test' in the table name.
    # tables = ["~^test.*"]
```

```
    # The following configuration checks all the tables in the database.
    # tables = ["~^"]

#### Special configuration for some tables
#### The configured table must be included in "check-tables'.
[[table-config]]
    # The name of the schema in the target database
    schema = "test"

    # The table name
    table = "test3"

    # Specifies the column used to divide data into chunks. If you do not
       ↪ configure it,
    # sync-diff-inspector chooses an appropriate column (primary key, unique
       ↪  key, or a field with index).
    index-field = "id"

    # Specifies the range of the data to be checked
    # It needs to comply with the syntax of the WHERE clause in SQL.
    range = "age > 10 AND age < 20"

    # Sets it to "true" when comparing the data of multiple sharded tables
    # with the data of the combined table.
    is-sharding = false

    # The collation of the string type of data might be inconsistent in some
       ↪  conditions.
    # You can specify "collation" to guarantee the order consistency.
    # You need to keep it corresponding to the "charset" setting in the
       ↪ database.
    # collation = "latin1_bin"

    # Ignores checking some columns such as some types (json, bit, blob, etc
       ↪ .)
    # that sync-diff-inspector does not currently support.
    # The floating-point data type behaves differently in TiDB and MySQL.
       ↪ You can use
    # `ignore-columns` to skip checking these columns
    # ignore-columns = ["name"]

#### Configuration example of comparing two tables with different schema
   ↪ names and table names.
[[table-config]]
    # The name of the target schema
```

```
    schema = "test"

    # The name of the target table
    table = "test2"

    # Sets it to "false" in non-sharding scenarios.
    is-sharding = false

    # Configuration of the source data
    [[table-config.source-tables]]
        # The instance ID of the source schema
        instance-id = "source-1"
        # The name of the source schema
        schema = "test"
        # The name of the source table
        table = "test1"

############################## Databases config ##########################

#### Configuration of the source database instance
[[source-db]]
    host = "127.0.0.1"
    port = 3306
    user = "root"
    password = "123456"
   # The instance ID of the source database, the unique identifier of a
      ↪ database instance
    instance-id = "source-1"
    # Uses the snapshot function of TiDB.
    # If enabled, the history data is used for comparison.
    # snapshot = "2016-10-08 16:45:26"
    # Sets the `sql-mode` of the database to parse table structures.
    # sql-mode = ""

#### Configuration of the target database instance
[target-db]
    host = "127.0.0.1"
    port = 4000
    user = "root"
    password = "123456"
    # Uses the snapshot function of TiDB.
    # If enabled, the history data is used for comparison.
    # snapshot = "2016-10-08 16:45:26"
    # Sets the `sql-mode` of the database to parse table structures.
    # sql-mode = ""
```

Run sync-diff-inspector

Run the following command:

```
./bin/sync_diff_inspector --config=./config.toml
```

This command outputs a check report to the log and describes the check result of each table. sync-diff-inspector generates the SQL statements to fix inconsistent data and stores these statements in a `fix.sql` file.

Note

- sync-diff-inspector consumes a certain amount of server resources when checking data. Avoid using sync-diff-inspector to check data during peak business hours.
- TiDB uses the `utf8_bin` collation. If you need to compare the data in MySQL with that in TiDB, pay attention to the collation configuration of MySQL tables. If the primary key or unique key is the `varchar` type and the collation configuration in MySQL differs from that in TiDB, then the final check result might be incorrect because of the collation issue. You need to add collation to the sync-diff-inspector configuration file.
- sync-diff-inspector divides data into chunks first according to TiDB statistics and you need to guarantee the accuracy of the statistics. You can manually run the `analyze table {table_name}` command when the TiDB server's *workload is light*.
- Pay special attention to `table-rules`. If you configure `schema-pattern="test1"` and `target-schema="test2"`, the `test1` schema in the source database and the `test2` schema in the target database are compared. If the source database has a `test2` schema, this `test2` schema is also compared with the `test2` schema in the target database.
- The generated `fix.sql` is only used as a reference for repairing data, and you need to confirm it before executing these SQL statements to repair data.

4.16.10.2 Data Check for Tables with Different Schema or Table Names

When using replication tools such as TiDB Data Migration, you can set `route-rules` to replicate data to a specified table in the downstream. sync-diff-inspector enables you to verify tables with different schema names or table names.

Below is a simple example.

```
############################## Tables config ##########################

#### Configure the tables of the target database that need to be checked
[[check-tables]]
    # The name of the schema in the target database
    schema = "test_2"
```

```
    # The table that needs to be checked
    tables = ["t_2"]

#### Configuration example of comparing two tables with different schema
    ↪ names and table names
[[table-config]]
    # The name of the schema in the target database
    schema = "test_2"

    # The name of the target table
    table = "t_2"

    # Configuration of the source data
    [[table-config.source-tables]]
        # The instance ID of the source schema
        instance-id = "source-1"
        # The name of the source schema
        schema = "test_1"
        # The name of the source table
        table = "t_1"
```

This configuration can be used to check `test_2.t_2` in the downstream and `test_1.t_1` in the `source-1` instance.

To check a large number of tables with different schema names or table names, you can simplify the configuration by setting the mapping relationship by using `table-rule`. You can configure the mapping relationship of either schema or table, or of both. For example, all the tables in the upstream `test_1` database are replicated to the downstream `test_2` database, which can be checked through the following configuration:

```
############################## Tables config ##########################

#### Configures the tables of the target database that need to be checked
[[check-tables]]
    # The name of the schema in the target database
    schema = "test_2"

    # Check all the tables
    tables = ["~^"]

[[table-rules]]
    # schema-pattern and table-pattern support the wildcards "*" and "?"
    schema-pattern = "test_1"
    #table-pattern = ""
    target-schema = "test_2"
    #target-table = ""
```

4.16.10.3 Data Check in the Sharding Scenario

sync-diff-inspector supports data check in the sharding scenario. Assume that you use the DM replication tool to replicate data from multiple MySQL instances into TiDB, and now you can use sync-diff-inspector to check upstream and downstream data.

4.16.10.3.1 Use `table-config` for configuration

You can use `table-config` to configure `table-0`, set `is-sharding=true` and configure the upstream table information in `table-config.source-tables`. This configuration method requires setting all sharded tables, which is suitable for scenarios where the number of upstream sharded tables is small and the naming rules of sharded tables do not have a pattern as shown below.

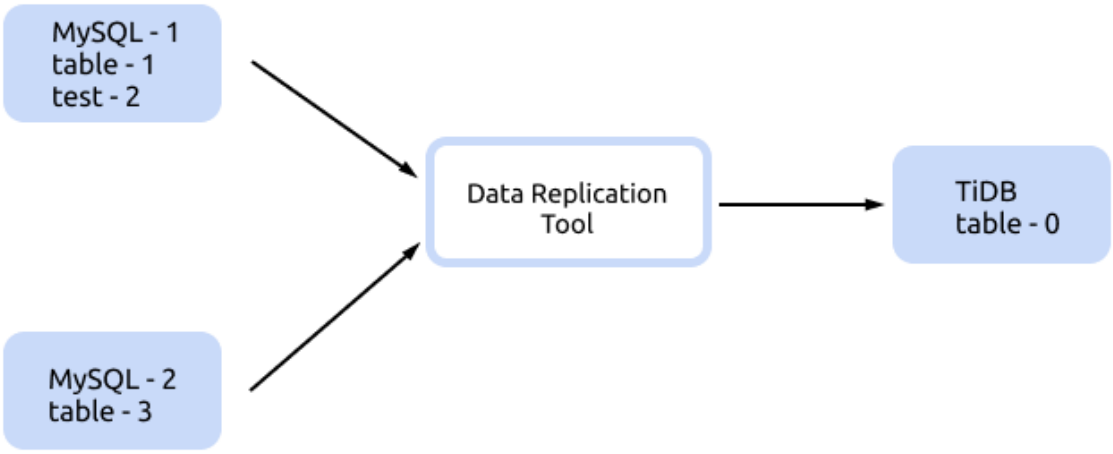

Figure 348: shard-table-replica-1

Below is a complete example of the sync-diff-inspector configuration.

```
#### Diff Configuration.

############################# Global config ############################

#### The log level. You can set it to "info" or "debug".
log-level = "info"

#### sync-diff-inspector divides the data into multiple chunks based on the
    ↪ primary key,
#### unique key, or the index, and then compares the data of each chunk.
#### Compares data in each chunk. Uses "chunk-size" to set the size of a
    ↪ chunk.
```

```
chunk-size = 1000

#### The number of goroutines created to check data
check-thread-count = 4

#### The proportion of sampling check. If you set it to 100, all the data is
    ↪  checked.
sample-percent = 100

#### If enabled, the chunk's checksum is calculated and data is compared by
    ↪ checksum.
#### If disabled, data is compared line by line.
use-checksum = true

#### If it is set to true, data is checked only by calculating checksum.
    ↪ Data is not checked after inspection, even if the upstream and
    ↪ downstream checksums are inconsistent.
only-use-checksum = false

#### Whether to use the checkpoint of the last check. If it is enabled, the
    ↪ inspector only checks the last unchecked chunks and chunks that
    ↪ failed the verification.
use-checkpoint = true

#### If it is set to true, data check is ignored.
#### If it is set to false, data is checked.
ignore-data-check = false

#### If it is set to true, the table struct comparison is ignored.
#### If set to false, the table struct is compared.
ignore-struct-check = false

#### The name of the file which saves the SQL statements used to repair data
fix-sql-file = "fix.sql"

############################## Tables config ##########################

#### Configures the tables of the target database that need to be checked
[[check-tables]]
    # The name of the schema in the target database
    schema = "test"

    # The name of tables that need to be checked in the target database
    tables = ["table-0"]
```

```
#### Configures the sharded tables corresponding to this table
[[table-config]]
    # The name of the target schema
    schema = "test"

    # The name of the table in the target schema
    table = "table-0"

    # Sets it to "true" in the sharding scenario
    is-sharding = true

    # Configuration of the source tables
    [[table-config.source-tables]]
    # The instance ID of the source database
    instance-id = "MySQL-1"
    schema = "test"
    table = "table-1"

    [[table-config.source-tables]]
    # The ID of the instance in the source database
    instance-id = "MySQL-1"
    schema = "test"
    table = "test-2"

    [[table-config.source-tables]]
    # The instance ID of the source database
    instance-id = "MySQL-2"
    schema = "test"
    table = "table-3"

############################## Databases config ##########################

#### Configuration of the source database instance
[[source-db]]
    host = "127.0.0.1"
    port = 3306
    user = "root"
    password = "123456"
    instance-id = "MySQL-1"

#### Configuration of the source database instance
[[source-db]]
    host = "127.0.0.2"
    port = 3306
    user = "root"
```

```
    password = "123456"
    instance-id = "MySQL-2"

#### Configuration of the target database instance
[target-db]
    host = "127.0.0.3"
    port = 4000
    user = "root"
    password = "123456"
    instance-id = "target-1"
```

4.16.10.3.2 Use `table-rules` for configuration

You can use `table-rules` for configuration when there are a large number of upstream sharded tables and the naming rules of all sharded tables have a pattern, as shown below:

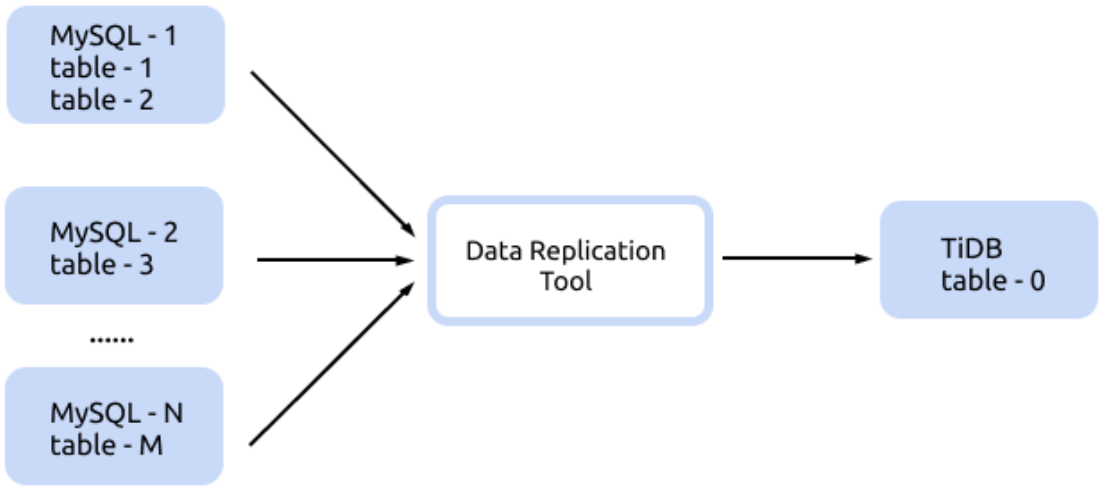

Figure 349: shard-table-replica-2

Below is a complete example of the sync-diff-inspector configuration.

```
#### Diff Configuration.
############################# Global config ############################
#### The log level. You can set it to "info" or "debug".
log-level = "info"

#### sync-diff-inspector divides the data into multiple chunks based on the
    ↪ primary key,
#### unique key, or the index, and then compares the data of each chunk.
```

```
#### Uses "chunk-size" to set the size of a chunk.
chunk-size = 1000

#### The number of goroutines created to check data
check-thread-count = 4

#### The proportion of sampling check. If you set it to 100, all the data is
    ↪  checked.
sample-percent = 100

#### If enabled, the chunk's checksum is calculated and data is compared by
    ↪ checksum.
#### If disabled, data is compared line by line.
use-checksum = true

#### If it is set to true, data is checked only by calculating checksum.
    ↪ Data is not checked after inspection, even if the upstream and
    ↪ downstream checksums are inconsistent.
only-use-checksum = false

#### Whether to use the checkpoint of the last check. If it is enabled, the
    ↪ inspector only checks the last unchecked chunks and chunks that
    ↪ failed the verification.
use-checkpoint = true

#### If it is set to true, data check is ignored.
#### If it is set to false, data is checked.
ignore-data-check = false

#### If it is set to true, the table struct comparison is ignored.
#### If set to false, the table struct is compared.
ignore-struct-check = false

#### The name of the file which saves the SQL statements used to repair data
fix-sql-file = "fix.sql"

############################## Tables config ##########################

#### Configures the tables of the target database that need to be checked
[[check-tables]]
    # The name of the schema in the target database
    schema = "test"

    # The name of tables that need to be checked in the target database
    tables = ["table-0"]
```

```
#### Use `table-rule` to set the mapping relationship between the upstream
    ↪ sharded tables and the downstream table family. You can configure the
    ↪  mapping rule only for the schema or table, or the mapping rules for
    ↪ both the schema and table.
[[table-rules]]
    # schema-pattern and table-pattern support wildcard *?
    # All tables that meet the schema-pattern and table-pattern rules in
       ↪ the upstream database configured in source-db are the sharded
       ↪ tables of target-schema.target-table.
    schema-pattern = "test"
    table-pattern = "table-*"
    target-schema = "test"
    target-table = "table-0"

############################## Databases config ##########################

#### Configuration of the source database instance
[[source-db]]
   host = "127.0.0.1"
   port = 3306
   user = "root"
   password = "123456"
   instance-id = "MySQL-1"

#### Configuration of the source database instance
[[source-db]]
   host = "127.0.0.2"
   port = 3306
   user = "root"
   password = "123456"
   instance-id = "MySQL-2"

#### Configuration of the target database instance
[target-db]
   host = "127.0.0.3"
   port = 4000
   user = "root"
   password = "123456"
   instance-id = "target-1"
```

4.16.10.4 Data Check for TiDB Upstream and Downstream Clusters

You can use TiDB Binlog to build upstream and downstream clusters of TiDB. When

Drainer replicates data to TiDB, the checkpoint is saved and the TSO mapping relationship between the upstream and downstream is also saved as `ts-map`. To check data between the upstream and downstream, configure `snapshot` in sync-diff-inspector.

4.16.10.4.1 Step 1: obtain `ts-map`

To obtain `ts-map`, execute the following SQL statement in the downstream TiDB cluster:

```
mysql> select * from tidb_binlog.checkpoint;
+--
    ↪ ------------------+-----------------------------------------------------------
    ↪
| clusterID         | checkPoint
    ↪
    ↪ |
+--
    ↪ ------------------+-----------------------------------------------------------
    ↪
| 6711243465327639221 | {"commitTS":409622383615541249,"ts-map":{"primary-ts
    ↪ ":409621863377928194,"secondary-ts":409621863377928345}} |
+--
    ↪ ------------------+-----------------------------------------------------------
    ↪
```

4.16.10.4.2 Step 2: configure snapshot

Then configure the snapshot information of the upstream and downstream databases by using the `ts-map` information obtained in Step 1.

Here is a configuration example of the `Databases config` section:

```
############################# Databases config ##########################

#### Configuration of the source database instance
[[source-db]]
    host = "127.0.0.1"
    port = 4000
    user = "root"
    password = "123456"
    # The instance ID of the source database, the unique identifier of a
        ↪ database instance
    instance-id = "source-1"
    # Uses the snapshot function of TiDB, corresponding to the primary-ts in
        ↪  ts-map
    snapshot = "409621863377928194"
```

```
#### Configuration of the target database instance
[target-db]
    host = "127.0.0.1"
    port = 4001
    user = "root"
    password = "123456"
    # Uses the snapshot function of TiDB, corresponding to the secondary-ts
        ↪ in ts-map
    snapshot = "409621863377928345"
```

Note:

- Set `db-type` of Drainer to `tidb` to ensure that `ts-map` is saved in the checkpoint.
- Modify the Garbage Collection (GC) time of TiKV to ensure that the historical data corresponding to snapshot is not collected by GC during the data check. It is recommended that you modify the GC time to 1 hour and recover the setting after the check.
- In some versions of TiDB Binlog, `master-ts` and `slave-ts` are stored in `ts-map`. `master-ts` is equivalent to `primary-ts` and `slave-ts` is equivalent to `secondary-ts`.

4.16.11 PD Control User Guide

As a command line tool of PD, PD Control obtains the state information of the cluster and tunes the cluster.

Note:

It is recommended that the version of the Control tool you use is consistent with the version of the cluster.

4.16.11.1 Compile from source code

1. Go Version 1.13 or later because the Go modules are used.
2. In the root directory of the PD project, use the `make` command to compile and generate `bin/pd-ctl`.

4.16.11.2 Download TiDB installation package

If you want to download the latest version of `pd-ctl`, directly download the TiDB package, because `pd-ctl` is included in the TiDB package.

Package download link	OS	Architecture	SHA256 checksum
`https://download.pingcap.org/tidb-{version}-linux-amd64.tar.gz` (pd-ctl)	Linux	amd64	`https://download.pingcap.org/tidb-{version}-linux-amd64.sha256`

Note:

`{version}` indicates the version number of TiDB. For example, if `{version}` is `v3.0.7`, the package download link is `https://download.pingcap.org/tidb-v3.0.7-linux-amd64.tar.gz`. You can also download the latest unpublished version by replacing `{version}` with `latest`.

4.16.11.3 Usage

Single-command mode:

```
./pd-ctl store -u http://127.0.0.1:2379
```

Interactive mode:

```
./pd-ctl -i -u http://127.0.0.1:2379
```

Use environment variables:

```
export PD_ADDR=http://127.0.0.1:2379
./pd-ctl
```

Use TLS to encrypt:

```
./pd-ctl -u https://127.0.0.1:2379 --cacert="path/to/ca" --cert="path/to/
    ↪ cert" --key="path/to/key"
```

4.16.11.4 Command line flags

4.16.11.4.1 –cacert

- Specifies the path to the certificate file of the trusted CA in PEM format
- Default: “”

4.16.11.4.2 –cert

- Specifies the path to the certificate of SSL in PEM format
- Default: “”

4.16.11.4.3 --detach,-d

- Uses the single command line mode (not entering readline)
- Default: true

4.16.11.4.4 --interact,-i

- Uses the interactive mode (entering readline)
- Default: false

4.16.11.4.5 –key

- Specifies the path to the certificate key file of SSL in PEM format, which is the private key of the certificate specified by `--cert`
- Default: “”

4.16.11.4.6 --pd,-u

- Specifies the PD address
- Default address: `http://127.0.0.1:2379`
- Environment variable: `PD_ADDR`

4.16.11.4.7 –version,-V

- Prints the version information and exit
- Default: false

4.16.11.5 Command

4.16.11.5.1 `cluster`

Use this command to view the basic information of the cluster.

Usage:

```
>> cluster                                  // To show the cluster information
{
  "id": 6493707687106161130,
  "max_peer_count": 3
}
```

4.16.11.5.2 `config [show | set <option> <value>]`

Use this command to view or modify the configuration information.

Usage:

```
>> config show                              // Display the config information
   ↪ of the scheduler
{
  "max-snapshot-count": 3,
  "max-pending-peer-count": 16,
  "max-merge-region-size": 50,
  "max-merge-region-keys": 200000,
  "split-merge-interval": "1h",
  "patrol-region-interval": "100ms",
  "max-store-down-time": "1h0m0s",
  "leader-schedule-limit": 4,
  "region-schedule-limit": 4,
  "replica-schedule-limit":8,
  "merge-schedule-limit": 8,
  "tolerant-size-ratio": 5,
```

```
  "low-space-ratio": 0.8,
  "high-space-ratio": 0.6,
  "disable-raft-learner": "false",
  "disable-remove-down-replica": "false",
  "disable-replace-offline-replica": "false",
  "disable-make-up-replica": "false",
  "disable-remove-extra-replica": "false",
  "disable-location-replacement": "false",
  "disable-namespace-relocation": "false",
  "schedulers-v2": [
    {
      "type": "balance-region",
      "args": null
    },
    {
      "type": "balance-leader",
      "args": null
    },
    {
      "type": "hot-region",
      "args": null
    }
  ]
}
>> config show all                          // Display all config information
>> config show namespace ts1                // Display the config information
   ↪ of the namespace named ts1
{
  "leader-schedule-limit": 4,
  "region-schedule-limit": 4,
  "replica-schedule-limit": 8,
  "max-replicas": 3,
}
>> config show replication                  // Display the config information
   ↪ of replication
{
  "max-replicas": 3,
  "location-labels": ""
}
>> config show cluster-version              // Display the current version of
   ↪ the cluster, which is the current minimum version of TiKV nodes in
   ↪ the cluster and does not correspond to the binary version.
"2.0.0"
```

- `max-snapshot-count` controls the maximum number of snapshots that a single store receives or sends out at the same time. The scheduler is restricted by this configuration to avoid taking up normal application resources. When you need to improve the speed of adding replicas or balancing, increase this value.

```
>> config set max-snapshot-count 16 // Set the maximum number of
   ↪ snapshots to 16
```

- `max-pending-peer-count` controls the maximum number of pending peers in a single store. The scheduler is restricted by this configuration to avoid producing a large number of Regions without the latest log in some nodes. When you need to improve the speed of adding replicas or balancing, increase this value. Setting it to 0 indicates no limit.

```
>> config set max-pending-peer-count 64 // Set the maximum number of
   ↪ pending peers to 64
```

- `max-merge-region-size` controls the upper limit on the size of Region Merge (the unit is M). When `regionSize` exceeds the specified value, PD does not merge it with the adjacent Region. Setting it to 0 indicates disabling Region Merge.

```
>> config set max-merge-region-size 16 // Set the upper limit on the
   ↪ size of Region Merge to 16M
```

- `max-merge-region-keys` controls the upper limit on the key count of Region Merge. When `regionKeyCount` exceeds the specified value, PD does not merge it with the adjacent Region.

```
>> config set max-merge-region-keys 50000 // Set the the upper limit on
   ↪  keyCount to 50000
```

- `split-merge-interval` controls the interval between the `split` and `merge` operations on a same Region. This means the newly split Region won't be merged within a period of time.

```
>> config set split-merge-interval 24h // Set the interval between `
   ↪ split` and `merge` to one day
```

- `patrol-region-interval` controls the execution frequency that `replicaChecker` checks the health status of Regions. A shorter interval indicates a higher execution frequency. Generally, you do not need to adjust it.

```
>> config set patrol-region-interval 50ms // Set the execution
   ↪ frequency of replicaChecker to 50ms
```

- `max-store-down-time` controls the time that PD decides the disconnected store cannot be restored if exceeded. If PD does not receive heartbeats from a store within the specified period of time, PD adds replicas in other nodes.

```
>> config set max-store-down-time 30m // Set the time within which PD
   ↪ receives no heartbeats and after which PD starts to add replicas
   ↪ to 30 minutes
```

- `leader-schedule-limit` controls the number of tasks scheduling the leader at the same time. This value affects the speed of leader balance. A larger value means a higher speed and setting the value to `0` closes the scheduling. Usually the leader scheduling has a small load, and you can increase the value in need.

```
>> config set leader-schedule-limit 4   // 4 tasks of leader scheduling
   ↪  at the same time at most
```

- `region-schedule-limit` controls the number of tasks scheduling the Region at the same time. This value affects the speed of Region balance. A larger value means a higher speed and setting the value to `0` closes the scheduling. Usually the Region scheduling has a large load, so do not set a too large value.

```
>> config set region-schedule-limit 2   // 2 tasks of Region scheduling
   ↪  at the same time at most
```

- `replica-schedule-limit` controls the number of tasks scheduling the replica at the same time. This value affects the scheduling speed when the node is down or removed. A larger value means a higher speed and setting the value to `0` closes the scheduling. Usually the replica scheduling has a large load, so do not set a too large value.

```
>> config set replica-schedule-limit 4  // 4 tasks of replica
   ↪ scheduling at the same time at most
```

- `merge-schedule-limit` controls the number of Region Merge scheduling tasks. Setting the value to `0` closes Region Merge. Usually the Merge scheduling has a large load, so do not set a too large value.

```
>> config set merge-schedule-limit 16 // 16 tasks of Merge scheduling
   ↪ at the same time at most
```

The configuration above is global. You can also tune the configuration by configuring different namespaces. The global configuration is used if the corresponding configuration of the namespace is not set.

Note:

The configuration of the namespace only supports editing `leader-schedule-limit`, `region-schedule-limit`, `replica-schedule-limit` and `max-replicas`.

```
>> config set namespace ts1 leader-schedule-limit 4 // 4 tasks of leader scheduling at the same time at most for the namespace named ts1
>> config set namespace ts2 region-schedule-limit 2 // 2 tasks of region scheduling at the same time at most for the namespace named ts2
```

- `tolerant-size-ratio` controls the size of the balance buffer area. When the score difference between the leader or Region of the two stores is less than specified multiple times of the Region size, it is considered in balance by PD.

```
>> config set tolerant-size-ratio 20   // Set the size of the buffer area to about 20 times of the average regionSize
```

- `low-space-ratio` controls the threshold value that is considered as insufficient store space. When the ratio of the space occupied by the node exceeds the specified value, PD tries to avoid migrating data to the corresponding node as much as possible. At the same time, PD mainly schedules the remaining space to avoid using up the disk space of the corresponding node.

```
config set low-space-ratio 0.9         // Set the threshold value of insufficient space to 0.9
```

- `high-space-ratio` controls the threshold value that is considered as sufficient store space. When the ratio of the space occupied by the node is less than the specified value, PD ignores the remaining space and mainly schedules the actual data volume.

```
config set high-space-ratio 0.5        // Set the threshold value of sufficient space to 0.5
```

- `disable-raft-learner` is used to disable Raft learner. By default, PD uses Raft learner when adding replicas to reduce the risk of unavailability due to downtime or network failure.

```
config set disable-raft-learner true   // Disable Raft learner
```

- `cluster-version` is the version of the cluster, which is used to enable or disable some features and to deal with the compatibility issues. By default, it is the minimum version of all normally running TiKV nodes in the cluster. You can set it manually only when you need to roll it back to an earlier version.

```
config set cluster-version 1.0.8        // Set the version of the
    ↪ cluster to 1.0.8
```

- `disable-remove-down-replica` is used to disable the feature of automatically deleting DownReplica. When you set it to `true`, PD does not automatically clean up the downtime replicas.
- `disable-replace-offline-replica` is used to disable the feature of migrating OfflineReplica. When you set it to `true`, PD does not migrate the offline replicas.
- `disable-make-up-replica` is used to disable the feature of making up replicas. When you set it to `true`, PD does not adding replicas for Regions without sufficient replicas.
- `disable-remove-extra-replica` is used to disable the feature of removing extra replicas. When you set it to `true`, PD does not remove extra replicas for Regions with redundant replicas.
- `disable-location-replacement` is used to disable the isolation level check. When you set it to `true`, PD does not improve the isolation level of Region replicas by scheduling.
- `disable-namespace-relocation` is used to disable Region relocation to the store of its namespace. When you set it to `true`, PD does not move Regions to stores where they belong to.

4.16.11.5.3 `config delete namespace <name> [<option>]`

Use this command to delete the configuration of namespace.

Usage:

After you configure the namespace, if you want it to continue to use global configuration, delete the configuration information of the namespace using the following command:

```
>> config delete namespace ts1                 // Delete the configuration of
    ↪  the namespace named ts1
```

If you want to use global configuration only for a certain configuration of the namespace, use the following command:

```
>> config delete namespace region-schedule-limit ts2 // Delete the region-
    ↪ schedule-limit configuration of the namespace named ts2
```

4.16.11.5.4 health

Use this command to view the health information of the cluster.

Usage:

```
>> health                              // Display the health information
[
  {
    "name": "pd",
    "member_id": 13195394291058371180,
    "client_urls": [
      "http://127.0.0.1:2379"
      ......
    ],
    "health": true
  }
  ......
]
```

4.16.11.5.5 hot [read | write | store]

Use this command to view the hot spot information of the cluster.

Usage:

```
>> hot read                            // Display hot spot for the read
   ↪ operation
>> hot write                           // Display hot spot for the write
   ↪ operation
>> hot store                           // Display hot spot for all the read and
   ↪  write operations
```

4.16.11.5.6 label [store <name> <value>]

Use this command to view the label information of the cluster.

Usage:

```
>> label                               // Display all labels
>> label store zone cn                 // Display all stores including the "zone
   ↪ ":"cn" label
```

4.16.11.5.7 member [delete | leader_priority | leader [show | resign | transfer <member_name>]]

Use this command to view the PD members, remove a specified member, or configure the priority of leader.

Usage:

```
>> member                              // Display the information of all members
{
  "members": [......],
  "leader": {......},
  "etcd_leader": {......},
}
>> member delete name pd2              // Delete "pd2"
Success!
>> member delete id 1319539429105371180 // Delete a node using id
Success!
>> member leader show                  // Display the leader information
{
  "name": "pd",
  "addr": "http://192.168.199.229:2379",
  "id": 9724873857558226554
}
>> member leader resign // Move leader away from the current member
......
>> member leader transfer pd3 // Migrate leader to a specified member
......
```

4.16.11.5.8 operator [show | add | remove]

Use this command to view and control the scheduling operation, split a Region, or merge Regions.

Usage:

```
>> operator show                                  // Display all operators
>> operator show admin                            // Display all admin
   ↪ operators
>> operator show leader                           // Display all leader
   ↪ operators
>> operator show region                           // Display all Region
   ↪ operators
>> operator add add-peer 1 2                      // Add a replica of Region
   ↪ 1 on store 2
>> operator add remove-peer 1 2                   // Remove a replica of
   ↪ Region 1 on store 2
>> operator add transfer-leader 1 2               // Schedule the leader of
   ↪ Region 1 to store 2
>> operator add transfer-region 1 2 3 4           // Schedule Region 1 to
   ↪ stores 2,3,4
```

```
>> operator add transfer-peer 1 2 3              // Schedule the replica of
   ↪ Region 1 on store 2 to store 3
>> operator add merge-region 1 2                 // Merge Region 1 with
   ↪ Region 2
>> operator add split-region 1 --policy=approximate // Split Region 1 into
   ↪ two Regions in halves, based on approximately estimated value
>> operator add split-region 1 --policy=scan     // Split Region 1 into two
   ↪ Regions in halves, based on accurate scan value
>> operator remove 1                             // Remove the scheduling
   ↪ operation of Region 1
```

The splitting of Regions starts from the position as close as possible to the middle. You can locate this position using two strategies, namely "scan" and "approximate". The difference between them is that the former determines the middle key by scanning the Region, and the latter obtains the approximate position by checking the statistics recorded in the SST file. Generally, the former is more accurate, while the latter consumes less I/O and can be completed faster.

4.16.11.5.9 `ping`

Use this command to view the time that `ping` PD takes.

Usage:

```
>> ping
time: 43.12698ms
```

4.16.11.5.10 `region <region_id> [--jq="<query string>"]`

Use this command to view the region information. For a jq formatted output, see jq-formatted-json-output-usage.

Usage:

```
>> region                           //  Display the information of all
   ↪ regions
{
  "count": 1,
  "regions": [......]
}

>> region 2                         // Display the information of the region
   ↪ with the id of 2
{
  "region": {
     "id": 2,
     ......
```

```
  }
  "leader": {
      ......
  }
}
```

4.16.11.5.11 region key [--format=raw|encode] <key>

Use this command to query the region that a specific key resides in. It supports the raw and encoding formats. And you need to use single quotes around the key when it is in the encoding format.

Raw format usage (default):

```
>> region key abc
{
  "region": {
    "id": 2,
    ......
  }
}
```

Encoding format usage:

```
>> region key --format=encode 't\200\000\000\000\000\000\000\377\035_r
   ↪ \200\000\000\000\000\377\017U\320\000\000\000\000\000\000\372'
{
  "region": {
    "id": 2,
    ......
  }
}
```

4.16.11.5.12 region sibling <region_id>

Use this command to check the adjacent Regions of a specific Region.

Usage:

```
>> region sibling 2
{
  "count": 2,
  "regions": [......],
}
```

4.16.11.5.13 `region store <store_id>`

Use this command to list all Regions of a specific store.

Usage:

```
>> region store 2
{
  "count": 10,
  "regions": [......],
}
```

4.16.11.5.14 `region topread [limit]`

Use this command to list Regions with top read flow. The default value of the limit is 16.

Usage:

```
>> region topread
{
  "count": 16,
  "regions": [......],
}
```

4.16.11.5.15 `region topwrite [limit]`

Use this command to list Regions with top write flow. The default value of the limit is 16.

Usage:

```
>> region topwrite
{
  "count": 16,
  "regions": [......],
}
```

4.16.11.5.16 `region topconfver [limit]`

Use this command to list Regions with top conf version. The default value of the limit is 16.

Usage:

```
>> region topconfver
{
  "count": 16,
  "regions": [......],
```

```
}
```

4.16.11.5.17 region topversion [limit]

Use this command to list Regions with top version. The default value of the limit is 16.

Usage:

```
>> region topversion
{
  "count": 16,
  "regions": [......],
}
```

4.16.11.5.18 region topsize [limit]

Use this command to list Regions with top approximate size. The default value of the limit is 16.

Usage:

```
>> region topsize
{
   "count": 16,
   "regions": [......],
}
```

4.16.11.5.19 region check [miss-peer | extra-peer | down-peer | pending-peer | incorrect-ns]

Use this command to check the Regions in abnormal conditions.

Description of various types:

- miss-peer: the Region without enough replicas
- extra-peer: the Region with extra replicas
- down-peer: the Region in which some replicas are Down
- pending-peer: the Region in which some replicas are Pending
- incorrect-ns: the Region in which some replicas deviate from the namespace constraints

Usage:

```
>> region check miss-peer
{
  "count": 2,
  "regions": [......],
}
```

4.16.11.5.20 scheduler [show | add | remove]

Use this command to view and control the scheduling strategy.

Usage:

```
>> scheduler show                          // Display all schedulers
>> scheduler add grant-leader-scheduler 1 // Schedule all the leaders of the
   ↪  regions on store 1 to store 1
>> scheduler add evict-leader-scheduler 1 // Move all the region leaders on
   ↪ store 1 out
>> scheduler add shuffle-leader-scheduler // Randomly exchange the leader on
   ↪  different stores
>> scheduler add shuffle-region-scheduler // Randomly scheduling the regions
   ↪  on different stores
>> scheduler remove grant-leader-scheduler-1 // Remove the corresponding
   ↪ scheduler
```

4.16.11.5.21 store [delete | label | weight] <store_id> [--jq="<query string>"]

Use this command to view the store information or remove a specified store. For a jq formatted output, see jq-formatted-json-output-usage.

Usage:

```
>> store                    // Display information of all stores
{
  "count": 3,
  "stores": [...]
}
>> store 1                  // Get the store with the store id of 1
  ......
>> store delete 1           // Delete the store with the store id of 1
  ......
>> store label 1 zone cn    // Set the value of the label with the "zone"
   ↪ key to "cn" for the store with the store id of 1
>> store weight 1 5 10      // Set the leader weight to 5 and region weight
   ↪ to 10 for the store with the store id of 1
```

4.16.11.5.22 table_ns [create | add | remove | set_store | rm_store | set_meta | rm_meta]

Use this command to view the namespace information of the table.

Usage:

```
>> table_ns add ts1 1        // Add the table with the table id of 1 to the
   ↪ namespace named ts1
>> table_ns create ts1       // Add the namespace named ts1
>> table_ns remove ts1 1     // Remove the table with the table id of 1 from
   ↪  the namespace named ts1
>> table_ns rm_meta ts1      // Remove the metadata from the namespace named
   ↪  ts1
>> table_ns rm_store 1 ts1   // Remove the table with the store id of 1 from
   ↪  the namespace named ts1
>> table_ns set_meta ts1     // Add the metadata to namespace named ts1
>> table_ns set_store 1 ts1  // Add the table with the store id of 1 to the
   ↪ namespace named ts1
```

4.16.11.5.23 `tso`

Use this command to parse the physical and logical time of TSO.

Usage:

```
>> tso 395181938313123110    // Parse TSO
system: 2017-10-09 05:50:59 +0800 CST
logic: 120102
```

4.16.11.6 Jq formatted JSON output usage

4.16.11.6.1 Simplify the output of `store`

```
» store --jq=".stores[].store | { id, address, state_name}"
{"id":1,"address":"127.0.0.1:20161","state_name":"Up"}
{"id":30,"address":"127.0.0.1:20162","state_name":"Up"}
...
```

4.16.11.6.2 Query the remaining space of the node

```
» store --jq=".stores[] | {id: .store.id, available: .status.available}"
{"id":1,"available":"10 GiB"}
{"id":30,"available":"10 GiB"}
...
```

4.16.11.6.3 Query the distribution status of the Region replicas

```
» region --jq=".regions[] | {id: .id, peer_stores: [.peers[].store_id]}"
{"id":2,"peer_stores":[1,30,31]}
{"id":4,"peer_stores":[1,31,34]}
...
```

4.16.11.6.4 Filter Regions according to the number of replicas

For example, to filter out all Regions whose number of replicas is not 3:

```
» region --jq=".regions[] | {id: .id, peer_stores: [.peers[].store_id] |
   ↪ select(length != 3)}"
{"id":12,"peer_stores":[30,32]}
{"id":2,"peer_stores":[1,30,31,32]}
```

4.16.11.6.5 Filter Regions according to the store ID of replicas

For example, to filter out all Regions that have a replica on store30:

```
» region --jq=".regions[] | {id: .id, peer_stores: [.peers[].store_id] |
   ↪ select(any(.==30))}"
{"id":6,"peer_stores":[1,30,31]}
{"id":22,"peer_stores":[1,30,32]}
...
```

You can also find out all Regions that have a replica on store30 or store31 in the same way:

```
» region --jq=".regions[] | {id: .id, peer_stores: [.peers[].store_id] |
   ↪ select(any(.==(30,31)))}"
{"id":16,"peer_stores":[1,30,34]}
{"id":28,"peer_stores":[1,30,32]}
{"id":12,"peer_stores":[30,32]}
...
```

4.16.11.6.6 Look for relevant Regions when restoring data

For example, when [store1, store30, store31] is unavailable at its downtime, you can find all Regions whose Down replicas are more than normal replicas:

```
» region --jq=".regions[] | {id: .id, peer_stores: [.peers[].store_id] |
   ↪ select(length as $total | map(if .==(1,30,31) then . else empty end)
   ↪ | length>=$total-length) }"
{"id":2,"peer_stores":[1,30,31,32]}
{"id":12,"peer_stores":[30,32]}
{"id":14,"peer_stores":[1,30,32]}
...
```

Or when [store1, store30, store31] fails to start, you can find Regions where the data can be manually removed safely on store1. In this way, you can filter out all Regions that have a replica on store1 but don't have other DownPeers:

```
» region --jq=".regions[] | {id: .id, peer_stores: [.peers[].store_id] |
    ↪ select(length>1 and any(.==1) and all(.!=(30,31)))}"
{"id":24,"peer_stores":[1,32,33]}
```

When [store30, store31] is down, find out all Regions that can be safely processed by creating the `remove-peer` Operator, that is, Regions with one and only DownPeer:

```
» region --jq=".regions[] | {id: .id, remove_peer: [.peers[].store_id] |
    ↪ select(length>1) | map(if .==(30,31) then . else empty end) | select(
    ↪ length==1)}"
{"id":12,"remove_peer":[30]}
{"id":4,"remove_peer":[31]}
{"id":22,"remove_peer":[30]}
...
```

4.16.12 PD Recover User Guide

PD Recover is a disaster recovery tool of PD, used to recover the PD cluster which cannot start or provide services normally. PD Recover is downloaded with TiDB Ansible in the `resource/bin/pd-recover` path.

4.16.12.1 Quick Start

This section describes how to use PD Recover to recover a PD cluster.

4.16.12.1.1 Get cluster ID

The cluster ID can be obtained from the log of PD, TiKV or TiDB. To get the cluster ID, you can either use the `ansible ad-hoc` command in the Control Machine, or view the log directly on the server.

Get `[info] cluster ID` from PD log (recommended)

To get the `[info] cluster id` from the PD log, run the following command:

```
ansible -i inventory.ini pd_servers -m shell -a 'cat {{deploy_dir}}/log/pd.
    ↪ log | grep "init cluster id" | head -10'
```

```
10.0.1.13 | CHANGED | rc=0 >>
[2019/10/14 10:35:38.880 +00:00] [INFO] [server.go:212] ["init cluster id"]
    ↪ [cluster-id=6747551640615446306]
.....
```

Get `[info] cluster ID` from TiDB log

To get the `[info] cluster ID` from the TiDB log, run the following command:

```
ansible -i inventory.ini tidb_servers -m shell -a 'cat {{deploy_dir}}/log/
    ↪ tidb*.log | grep "init cluster id" | head -10'
```

```
10.0.1.15 | CHANGED | rc=0 >>
2019/10/14 19:23:04.688 client.go:161: [info] [pd] init cluster id
    ↪ 6747551640615446306
......
```

Get `[info] PD cluster` from TiKV log

To get the `[info] PD cluster` from the TiKV log, run the following command:

```
ansible -i inventory.ini tikv_servers -m shell -a 'cat {{deploy_dir}}/log/
    ↪ tikv* | grep "PD cluster" | head -10'
```

```
10.0.1.15 | CHANGED | rc=0 >>
[2019/10/14 07:06:35.278 +00:00] [INFO] [tikv-server.rs:464] ["connect to PD
    ↪  cluster 6747551640615446306"]
......
```

4.16.12.1.2 Get `Alloc ID` (TiKV StoreID)

The `alloc-id` value you specify must be larger than the currently largest `Alloc ID` value. To get `Alloc ID`, you can either use the `ansible ad-hoc` command in the Control Machine, or view the log directly on the server.

Get `[info] allocates id` from PD log

To get the `[info] allocates id` from the PD log, run the following command:

```
ansible -i inventory.ini pd_servers -m shell -a 'cat {{deploy_dir}}/log/pd*
    ↪ | grep "allocates" | head -10'
```

```
10.0.1.13 | CHANGED | rc=0 >>
[2019/10/15 03:15:05.824 +00:00] [INFO] [id.go:91] ["idAllocator allocates a
    ↪  new id"] [alloc-id=3000]
[2019/10/15 08:55:01.275 +00:00] [INFO] [id.go:91] ["idAllocator allocates a
    ↪  new id"] [alloc-id=4000]
......
```

Get `[info] alloc store id` from TiKV log

To get the `[info] alloc store id` from the TiKV log, run the following command:

```
ansible -i inventory.ini tikv_servers -m shell -a 'cat {{deploy_dir}}/log/
    ↪ tikv* | grep "alloc store" | head -10'
```

```
10.0.1.13 | CHANGED | rc=0 >>
[2019/10/14 07:06:35.516 +00:00] [INFO] [node.rs:229] ["alloc store id 4 "]
10.0.1.14 | CHANGED | rc=0 >>
[2019/10/14 07:06:35.734 +00:00] [INFO] [node.rs:229] ["alloc store id 5 "]
10.0.1.15 | CHANGED | rc=0 >>
[2019/10/14 07:06:35.418 +00:00] [INFO] [node.rs:229] ["alloc store id 1 "]
10.0.1.21 | CHANGED | rc=0 >>
[2019/10/15 03:15:05.826 +00:00] [INFO] [node.rs:229] ["alloc store id 2001
    ↪ "]
10.0.1.20 | CHANGED | rc=0 >>
[2019/10/15 03:15:05.987 +00:00] [INFO] [node.rs:229] ["alloc store id 2002
    ↪ "]
```

4.16.12.1.3 Deploy a new PD cluster

To deploy a new PD cluster, run the following command:

```
ansible-playbook bootstrap.yml --tags=pd &&
ansible-playbook deploy.yml --tags=pd &&
ansible-playbook start.yml --tags=pd
```

To delete the old cluster, delete the `data.pd` directory and restart the PD service.

4.16.12.1.4 Use pd-recover

```
./pd-recover -endpoints http://10.0.1.13:2379 -cluster-id
    ↪ 6747551640615446306 -alloc-id 10000
```

4.16.12.1.5 Restart PD cluster

```
ansible-playbook rolling_update.yml --tags=pd
```

4.16.12.1.6 Restart TiDB or TiKV

```
ansible-playbook rolling_update.yml --tags=tidb,tikv
```

4.16.12.2 FAQ

4.16.12.2.1 Multiple cluster IDs are found when getting the cluster ID

When a PD cluster is created, a new cluster ID is generated. You can determine the cluster ID of the old cluster by viewing the log.

4.16.12.2.2 The error `dial tcp 10.0.1.13:2379: connect: connection refused` is returned when executing `pd-recover`

The PD service is required when you execute `pd-recover`. Deploy and start the PD cluster before you use PD Recover.

4.16.13 TiKV Control User Guide

TiKV Control (`tikv-ctl`) is a command line tool of TiKV, used to manage the cluster. Its installation directory is as follows:

- If the cluster is deployed using TiDB Ansible, `tikv-ctl` directory is in the `resources` ↪ `/bin` subdirectory under the `ansible` directory.
- If the cluster is deployed using TiUP, `tikv-ctl` directory is in the in `~/.tiup/` ↪ `components/ctl/{VERSION}/` directory.

> **Note:**
>
> It is recommended that the version of the Control tool you use is consistent with the version of the cluster.

TiUP is a deployment tool introduced later than TiDB Ansible, and its usage is simpler. `tikv-ctl` is also integrated in the `tiup` command. Execute the following command to call the `tikv-ctl` tool:

```
tiup ctl tikv
```

```
Starting component `ctl`: ~/.tiup/components/ctl/v4.0.0-rc.2/ctl tikv
TiKV Control (tikv-ctl)
Release Version: 4.0.0-rc.2
Edition:         Community
Git Commit Hash: 2fdb2804bf8ffaab4b18c4996970e19906296497
Git Commit Branch: heads/refs/tags/v4.0.0-rc.2
UTC Build Time:  2020-05-15 11:58:49
Rust Version:    rustc 1.42.0-nightly (0de96d37f 2019-12-19)
Enable Features: jemalloc portable sse protobuf-codec
Profile:         dist_release
```

You can add corresponding parameters and subcommands after `tiup ctl tikv`.

4.16.13.1 General options

`tikv-ctl` provides two operation modes:

- Remote mode: use the `--host` option to accept the service address of TiKV as the argument

  For this mode, if SSL is enabled in TiKV, `tikv-ctl` also needs to specify the related certificate file. For example:

```
$ tikv-ctl --ca-path ca.pem --cert-path client.pem --key-path client-
    ↪ key.pem --host 127.0.0.1:20160 <subcommands>
```

  However, sometimes `tikv-ctl` communicates with PD instead of TiKV. In this case, you need to use the `--pd` option instead of `--host`. Here is an example:

```
$ tikv-ctl --pd 127.0.0.1:2379 compact-cluster
store:"127.0.0.1:20160" compact db:KV cf:default range:([], []) success
    ↪ !
```

- Local mode: Use the `--db` option to specify the local TiKV data directory path. In this mode, you need to stop the running TiKV instance.

Unless otherwise noted, all commands support both the remote mode and the local mode.

Additionally, `tikv-ctl` has two simple commands `--to-hex` and `--to-escaped`, which are used to make simple changes to the form of the key.

Generally, use the `escaped` form of the key. For example:

```
$ tikv-ctl --to-escaped 0xaaff
\252\377
$ tikv-ctl --to-hex "\252\377"
AAFF
```

> **Note:**
>
> When you specify the `escaped` form of the key in a command line, it is required to enclose it in double quotes. Otherwise, bash eats the backslash and a wrong result is returned.

4.16.13.2 Subcommands, some options and flags

This section describes the subcommands that `tikv-ctl` supports in detail. Some subcommands support a lot of options. For all details, run `tikv-ctl --help <subcommand>`.

4.16.13.2.1 View information of the Raft state machine

Use the `raft` subcommand to view the status of the Raft state machine at a specific moment. The status information includes two parts: three structs (**RegionLocalState**, **RaftLocalState**, and **RegionApplyState**) and the corresponding Entries of a certain piece of log.

Use the `region` and `log` subcommands to obtain the above information respectively. The two subcommands both support the remote mode and the local mode at the same time. Their usage and output are as follows:

```
$ tikv-ctl --host 127.0.0.1:20160 raft region -r 2
region id: 2
region state key: \001\003\000\000\000\000\000\000\000\002\001
region state: Some(region {id: 2 region_epoch {conf_ver: 3 version: 1} peers
    ↪ {id: 3 store_id: 1} peers {id: 5 store_id: 4} peers {id: 7 store_id:
    ↪ 6}})
raft state key: \001\002\000\000\000\000\000\000\000\002\002
raft state: Some(hard_state {term: 307 vote: 5 commit: 314617} last_index:
    ↪ 314617)
apply state key: \001\002\000\000\000\000\000\000\000\002\003
apply state: Some(applied_index: 314617 truncated_state {index: 313474 term:
    ↪ 151})
```

4.16.13.2.2 View the Region size

Use the `size` command to view the Region size:

```
$ tikv-ctl --db /path/to/tikv/db size -r 2
region id: 2
cf default region size: 799.703 MB
cf write region size: 41.250 MB
cf lock region size: 27616
```

4.16.13.2.3 Scan to view MVCC of a specific range

The `--from` and `--to` options of the `scan` command accept two escaped forms of raw key, and use the `--show-cf` flag to specify the column families that you need to view.

```
$ tikv-ctl --db /path/to/tikv/db scan --from 'zm' --limit 2 --show-cf lock,
    ↪ default,write
key: zmBootstr\377a\377pKey
    ↪ \000\000\377\000\000\373\000\000\000\000\000\377\000\000s
    ↪ \000\000\000\000\000\372
         write cf value: start_ts: 399650102814441473 commit_ts:
             ↪ 399650102814441475 short_value: "20"
```

```
key: zmDB:29\000\000\377\000\374\000\000\000\000\000\000\377\000H
    ↪ \000\000\000\000\000\000\371
        write cf value: start_ts: 399650105239273474 commit_ts:
            ↪ 399650105239273475 short_value: "
            ↪ \000\000\000\000\000\000\000\002"
        write cf value: start_ts: 399650105199951882 commit_ts:
            ↪ 399650105213059076 short_value: "
            ↪ \000\000\000\000\000\000\000\001"
```

4.16.13.2.4 View MVCC of a given key

Similar to the `scan` command, the `mvcc` command can be used to view MVCC of a given key.

```
$ tikv-ctl --db /path/to/tikv/db mvcc -k "zmDB
    ↪ :29\000\000\377\000\374\000\000\000\000\000\000\377\000H
    ↪ \000\000\000\000\000\000\371" --show-cf=lock,write,default
key: zmDB:29\000\000\377\000\374\000\000\000\000\000\000\377\000H
    ↪ \000\000\000\000\000\000\371
        write cf value: start_ts: 399650105239273474 commit_ts:
            ↪ 399650105239273475 short_value: "
            ↪ \000\000\000\000\000\000\000\002"
        write cf value: start_ts: 399650105199951882 commit_ts:
            ↪ 399650105213059076 short_value: "
            ↪ \000\000\000\000\000\000\000\001"
```

In this command, the key is also the escaped form of raw key.

4.16.13.2.5 Scan raw keys

The `raw-scan` command scans directly from the RocksDB. Note that to scan data keys you need to add a `'z'` prefix to keys.

Use `--from` and `--to` options to specify the range to scan (unbounded by default). Use `--limit` to limit at most how many keys to print out (30 by default). Use `--cf` to specify which cf to scan (can be `default`, `write` or `lock`).

```
$ ./tikv-ctl --db /var/lib/tikv/db/ raw-scan --from 'zt' --limit 2 --cf
    ↪ default
key: "zt\200\000\000\000\000\000\000\377\005_r
    ↪ \200\000\000\000\000\377\000\000\001\000\000\000\000\000\372\372b2
    ↪ ,^\033\377\364", value: "\010\002\002\002%\010\004\002\010root
    ↪ \010\006\002\000\010\010\t\002\010\n\t\002\010\014\t\002\010\016\t
    ↪ \002\010\020\t\002\010\022\t\002\010\024\t\002\010\026\t\002\010\030\
    ↪ t\002\010\032\t\002\010\034\t\002\010\036\t\002\010 \t\002\010\"\t
    ↪ \002\010s\t\002\010&\t\002\010(\t\002\010*\t\002\010,\t\002\010.\t
    ↪ \002\0100\t\002\0102\t\002\0104\t\002"
```

```
key: "zt\200\000\000\000\000\000\000\377\025_r
    ↪ \200\000\000\000\000\377\000\000\023\000\000\000\000\000\372\372b2
    ↪ ,^\033\377\364", value: "\010\002\002&slow_query_log_file\010\004\002
    ↪ P/usr/local/mysql/data/localhost-slow.log"

Total scanned keys: 2
```

4.16.13.2.6 Print a specific key value

To print the value of a key, use the `print` command.

4.16.13.2.7 Print some properties about Region

In order to record Region state details, TiKV writes some statistics into the SST files of Regions. To view these properties, run `tikv-ctl` with the `region-properties` subcommand:

```
$ tikv-ctl --host localhost:20160 region-properties -r 2
num_files: 0
num_entries: 0
num_deletes: 0
mvcc.min_ts: 18446744073709551615
mvcc.max_ts: 0
mvcc.num_rows: 0
mvcc.num_puts: 0
mvcc.num_versions: 0
mvcc.max_row_versions: 0
middle_key_by_approximate_size:
```

The properties can be used to check whether the Region is healthy or not. If not, you can use them to fix the Region. For example, splitting the Region manually by `middle_key_approximate_size`.

4.16.13.2.8 Compact data of each TiKV manually

Use the `compact` command to manually compact data of each TiKV. If you specify the `--from` and `--to` options, then their flags are also in the form of escaped raw key.

- Use the `--host` option to specify the TiKV that you need to compact.
- Use the `-d` option to specify the RocksDB that you need to compact. The optional values are `kv` and `raft`.
- Use the `--threads` option allows you to specify the concurrency that you compact and its default value is 8. Generally, a higher concurrency comes with a faster compact speed, which might yet affect the service. You need to choose an appropriate concurrency based on the scenario.

```
$ tikv-ctl --db /path/to/tikv/db compact -d kv
success!
```

4.16.13.2.9 Compact data of the whole TiKV cluster manually

Use the `compact-cluster` command to manually compact data of the whole TiKV cluster. The flags of this command have the same meanings and usage as those of the `compact` command.

4.16.13.2.10 Set a Region to tombstone

The `tombstone` command is usually used in circumstances where the sync-log is not enabled, and some data written in the Raft state machine is lost caused by power down.

In a TiKV instance, you can use this command to set the status of some Regions to tombstone. Then when you restart the instance, those Regions are skipped to avoid the restart failure caused by damaged Raft state machines of those Regions. Those Regions need to have enough healthy replicas in other TiKV instances to be able to continue the reads and writes through the Raft mechanism.

In general cases, you can remove the corresponding Peer of this Region using the `remove -peer` command:

```
pd-ctl operator add remove-peer <region_id> <store_id>
```

Then use the `tikv-ctl` tool to set a Region to tombstone on the corresponding TiKV instance to skip the health check for this Region at startup:

```
tikv-ctl --db /path/to/tikv/db tombstone -p 127.0.0.1:2379 -r <region_id>
```

```
success!
```

However, in some cases, you cannot easily remove this Peer of this Region from PD, so you can specify the `--force` option in `tikv-ctl` to forcibly set the Peer to tombstone:

```
tikv-ctl --db /path/to/tikv/db tombstone -p 127.0.0.1:2379 -r <region_id>,<
    ↪ region_id> --force
```

```
success!
```

Note:

- The `tombstone` command only supports the local mode.

- The argument of the `-p` option specifies the PD endpoints without the `http` prefix. Specifying the PD endpoints is to query whether PD can safely switch to Tombstone.

4.16.13.2.11 Send a `consistency-check` request to TiKV

Use the `consistency-check` command to execute a consistency check among replicas in the corresponding Raft of a specific Region. If the check fails, TiKV itself panics. If the TiKV instance specified by `--host` is not the Region leader, an error is reported.

```
$ tikv-ctl --host 127.0.0.1:20160 consistency-check -r 2
success!
$ tikv-ctl --host 127.0.0.1:20161 consistency-check -r 2
DebugClient::check_region_consistency: RpcFailure(RpcStatus { status:
    ↪ Unknown, details: Some("StringError(\"Leader is on store 1\")") })
```

Note:

- This command only supports the remote mode.
- Even if this command returns `success!`, you need to check whether TiKV panics. This is because this command is only a proposal that requests a consistency check for the leader, and you cannot know from the client whether the whole check process is successful or not.

4.16.13.2.12 Dump snapshot meta

This sub-command is used to parse a snapshot meta file at given path and print the result.

4.16.13.2.13 Print the Regions where the Raft state machine corrupts

To avoid checking the Regions while TiKV is started, you can use the `tombstone` command to set the Regions where the Raft state machine reports an error to Tombstone. Before running this command, use the `bad-regions` command to find out the Regions with errors, so as to combine multiple tools for automated processing.

```
$ tikv-ctl --db /path/to/tikv/db bad-regions
all regions are healthy
```

If the command is successfully executed, it prints the above information. If the command fails, it prints the list of bad Regions. Currently, the errors that can be detected include the mismatches between `last index`, `commit index` and `apply index`, and the loss of Raft log. Other conditions like the damage of snapshot files still need further support.

4.16.13.2.14 View Region properties

- To view in local the properties of Region 2 on the TiKV instance that is deployed in `/path/to/tikv`:

```
$ tikv-ctl --db /path/to/tikv/data/db region-properties -r 2
```

- To view online the properties of Region 2 on the TiKV instance that is running on `127.0.0.1:20160`:

```
$ tikv-ctl --host 127.0.0.1:20160 region-properties -r 2
```

4.16.13.2.15 Modify the RocksDB configuration of TiKV dynamically

You can use the `modify-tikv-config` command to dynamically modify the configuration arguments. Currently, it only supports dynamically modifying RocksDB related arguments.

- `-m` is used to specify the target RocksDB. You can set it to `kvdb` or `raftdb`.
- `-n` is used to specify the configuration name. You can refer to the arguments of `[rocksdb]` and `[raftdb]` (corresponding to `kvdb` and `raftdb`) in the TiKV configuration template. You can use `default|write|lock + . + argument name` to specify the configuration of different CFs. For `kvdb`, you can set it to `default`, `write`, or `lock`; for `raftdb`, you can only set it to `default`.
- `-v` is used to specify the configuration value.

```
$ tikv-ctl modify-tikv-config -m kvdb -n max_background_jobs -v 8
success !
$ tikv-ctl modify-tikv-config -m kvdb -n write.block-cache-size -v 256MB
success!
$ tikv-ctl modify-tikv-config -m raftdb -n default.disable_auto_compactions
    ↪ -v true
success!
```

4.16.13.2.16 Force Region to recover the service from failure of multiple replicas

Use the `unsafe-recover remove-fail-stores` command to remove the failed machines from the peer list of Regions. Then after you restart TiKV, these Regions can continue to

provide services using the other healthy replicas. This command is usually used in circumstances where multiple TiKV stores are damaged or deleted.

The `-s` option accepts multiple `store_id` separated by comma and uses the `-r` flag to specify involved Regions. Otherwise, all Regions' peers located on these stores will be removed by default.

```
$ tikv-ctl --db /path/to/tikv/db unsafe-recover remove-fail-stores -s 3 -r
    ↪ 1001,1002
success!
```

Note:

- This command only supports the local mode. It prints `success!` when successfully run.
- You must run this command for all stores where specified Regions' peers are located. If `-r` is not set, all Regions are involved, and you need to run this command for all stores.

4.16.13.2.17 Recover from MVCC data corruption

Use the `recover-mvcc` command in circumstances where TiKV cannot run normally caused by MVCC data corruption. It cross-checks 3 CFs ("default", "write", "lock") to recover from various kinds of inconsistency.

- Use the `-r` option to specify involved Regions by `region_id`.
- Use the `-p` option to specify PD endpoints.

```
$ tikv-ctl --db /path/to/tikv/db recover-mvcc -r 1001,1002 -p 127.0.0.1:2379
success!
```

Note:

- This command only supports the local mode. It prints `success!` when successfully run.
- The argument of the `-p` option specifies the PD endpoints without the `http` prefix. Specifying the PD endpoints is to query whether the specified `region_id` is validated or not.
- You need to run this command for all stores where specified Regions' peers are located.

4.16.13.2.18 Ldb Command

The `ldb` command line tool offers multiple data access and database administration commands. Some examples are listed below. For more information, refer to the help message displayed when running `tikv-ctl ldb` or check the documents from RocksDB.

Examples of data access sequence:

To dump an existing RocksDB in HEX:

```
$ tikv-ctl ldb --hex --db=/tmp/db dump
```

To dump the manifest of an existing RocksDB:

```
$ tikv-ctl ldb --hex manifest_dump --path=/tmp/db/MANIFEST-000001
```

You can specify the column family that your query is against using the `--column_family =<string>` command line.

`--try_load_options` loads the database options file to open the database. It is recommended to always keep this option on when the database is running. If you open the database with default options, the LSM-tree might be messed up, which cannot be recovered automatically.

4.16.14 TiDB Control User Guide

TiDB Control is a command-line tool of TiDB, usually used to obtain the status information of TiDB for debugging. This document introduces the features of TiDB Control and how to use these features.

4.16.14.1 Get TiDB Control

You can get TiDB Control by installing it using TiDB Ansible or by compiling it from source code.

> **Note:**
>
> It is recommended that the version of the Control tool you use is consistent with the version of the cluster.

4.16.14.1.1 Install TiDB Control using TiDB Ansible

For TiDB clusters deployed using TiDB Ansible, you can find the TiDB Control binary program `tidb-ctl` under the TiDB installation path.

4.16.14.1.2 Compile from source code

- Compilation environment requirement: Go Version 1.13 or later
- Compilation procedures: Go to the root directory of the TiDB Control project, use the `make` command to compile, and generate `tidb-ctl`.
- Compilation documentation: you can find the help files in the `doc` directory; if the help files are lost or you want to update them, use the `make doc` command to generate the help files.

4.16.14.2 Usage introduction

This section describes how to use commands, subcommands, options, and flags in `tidb-ctl`.

- command: characters without `-` or `--`
- subcommand: characters without `-` or `--` that follow a command
- option: characters with `-` or `--`
- flag: characters exactly following a command/subcommand or option, passing value to the command/subcommand or option

Usage example: `tidb-ctl schema in mysql -n db`

- `schema`: the command
- `in`: the subcommand of `schema`
- `mysql`: the flag of `in`
- `-n`: the option
- `db`: the flag of `-n`

Currently, TiDB Control has the following subcommands:

- `tidb-ctl base64decode`: used for `BASE64` decoding
- `tidb-ctl decoder`: used for `KEY` decoding
- `tidb-ctl etcd`: used for operating etcd
- `tidb-ctl log`: used to format the log file to expand the single-line stack information
- `tidb-ctl mvcc`: used to get the MVCC information
- `tidb-ctl region`: used to get the Region information
- `tidb-ctl schema`: used to get the schema information
- `tidb-ctl table`: used to get the table information

4.16.14.2.1 Get help

Use `tidb-ctl -h/--help` to get usage information.

TiDB Control consists of multiple layers of commands. You can use `-h/--help` after each command/subcommand to get its respective usage information.

The following example shows how to obtain the schema information:

Use `tidb-ctl schema -h` to get usage details. The `schema` command itself has two subcommands: `in` and `tid`.

- `in` is used to obtain the table schema of all tables in the database through the database name.
- `tid` is used to obtain the table schema by using the unique `table_id` in the whole database.

4.16.14.2.2 Global options

`tidb-ctl` has the following connection-related global options:

- `--host`: TiDB Service address (default 127.0.0.1)
- `--port`: TiDB status port (default 10080)
- `--pdhost`: PD Service address (default 127.0.0.1)
- `--pdport`: PD Service port (default 2379)

`--pdhost` and `--pdport` are mainly used in the `etcd` subcommand. For example, `tidb-ctl etcd ddlinfo`. If you do not specify the address and the port, the following default value is used:

- The default service address of TiDB and PD: `127.0.0.1`. The service address must be an IP address.
- The default service port of TiDB: `10080`.
- The default service port of PD: `2379`.

4.16.14.2.3 The `schema` command

The `in` subcommand

`in` is used to obtain the table schema of all tables in the database through the database name.

```
tidb-ctl schema in <database name>
```

For example, running `tidb-ctl schema in mysql` returns the following result:

```
[
    {
        "id": 13,
        "name": {
            "O": "columns_priv",
            "L": "columns_priv"
        },
             ...
        "update_timestamp": 399494726837600268,
        "ShardRowIDBits": 0,
        "Partition": null
    }
]
```

The result is displayed in the JSON format. (The above output is truncated.)

- If you want to specify the table name, use `tidb-ctl schema in <database> -n <table name>` to filter.

  For example, `tidb-ctl schema in mysql -n db` returns the table schema of the `db` table in the `mysql` database:

  ```
  {
      "id": 9,
      "name": {
          "O": "db",
          "L": "db"
      },
      ...
      "Partition": null
  }
  ```

  (The above output is also truncated.)

  If you do not want to use the default TiDB service address and port, use the `--host` and `--port` options to configure. For example, `tidb-ctl --host 172.16.55.88 --port 8898 schema in mysql -n db`.

The `tid` subcommand

`tid` is used to obtain the table schema by using the unique `table_id` in the whole database. You can use the `in` subcommand to get all table IDs of certain schema and use the `tid` subcommand to get the detailed table information.

For example, the table ID of `mysql.stat_meta` is `21`. You can use `tidb-ctl schema tid -i 21` to obtain the detail of `mysql.stat_meta`.

```
{
 "id": 21,
 "name": {
  "O": "stats_meta",
  "L": "stats_meta"
 },
 "charset": "utf8mb4",
 "collate": "utf8mb4_bin",
  ...
}
```

Like the `in` subcommand, if you do not want to use the default TiDB service address and status port, use the `--host` and `--port` options to specify the host and port.

The `base64decode` command

`base64decode` is used to decode `base64` data.

```
tidb-ctl base64decode [base64_data]
tidb-ctl base64decode [db_name.table_name] [base64_data]
tidb-ctl base64decode [table_id] [base64_data]
```

1. Execute the following SQL statement to prepare the environment:

```
use test;
create table t (a int, b varchar(20),c datetime default
    ↪ current_timestamp , d timestamp default current_timestamp,
    ↪ unique index(a));
insert into t (a,b,c) values(1,"哈哈 hello",NULL);
alter table t add column e varchar(20);
```

2. Obtian MVCC data using the HTTP API interface:

```
$ curl "http://$IP:10080/mvcc/index/test/t/a/1?a=1"
{
 "info": {
  "writes": [
   {
    "start_ts": 407306449994645510,
    "commit_ts": 407306449994645513,
    "short_value": "AAAAAAAAAAE=" # The unique index a stores the
        ↪ handle id of the corresponding row.
   }
  ]
 }
}%
```

```
$ curl "http://$IP:10080/mvcc/key/test/t/1"
{
 "info": {
  "writes": [
   {
    "start_ts": 407306588892692486,
    "commit_ts": 407306588892692489,
    "short_value": "CAIIAggEAhjlk4jlk4ggaGVsbG8IBgAICAmAgIDwjYuu0Rk=" #
       ↪  Row data that handle id is 1.
   }
  ]
 }
}%
```

3. Decode `handle id (uint64)` using `base64decode`.

```
$ tidb-ctl base64decode AAAAAAAAAAE=
hex: 0000000000000001
uint64: 1
```

4. Decode row data using `base64decode`.

```
$ ./tidb-ctl base64decode test.t
   ↪ CAIIAggEAhjlk4jlk4ggaGVsbG8IBgAICAmAgIDwjYuu0Rk=
a:     1
b:     哈哈 hello
c is NULL
d:     2019-03-28 05:35:30
e not found in data

# if the table id of test.t is 60, you can also use below command to do
   ↪  the same thing.
$ ./tidb-ctl base64decode 60
   ↪ CAIIAggEAhjlk4jlk4ggaGVsbG8IBgAICAmAgIDwjYuu0Rk=
a:     1
b:     哈哈 hello
c is NULL
d:     2019-03-28 05:35:30
e not found in data
```

4.16.14.2.4 The `decoder` command

- The following example shows how to decode the row key, similar to decoding the index key.

```
$ ./tidb-ctl decoder "t\x00\x00\x00\x00\x00\x00\x00\x1c_r\x00\x00\x00\
    ↪ x00\x00\x00\x00\xfa"
format: table_row
table_id: -9223372036854775780  table_id: -9223372036854775780
row_id: -9223372036854775558    row_id: -9223372036854775558
```

- The following example shows how to decode `value`.

```
$ ./tidb-ctl decoder AhZoZWxsbyB3b3JsZAiAEA==
format: index_value
type: bigint, value: 1024   index_value[0]: {type: bytes, value: hello
    ↪ world}
index_value[1]: {type: bigint, value: 1024}
```

4.16.14.2.5 The `etcd` command

- `tidb-ctl etcd ddlinfo` is used to obtain DDL information.
- `tidb-ctl etcd putkey KEY VALUE` is used to add KEY VALUE to etcd (All the KEYs are added to the `/tidb/ddl/all_schema_versions/` directory).

```
tidb-ctl etcd putkey "foo" "bar"
```

  In fact, a key-value pair is added to the etcd whose KEY is `/tidb/ddl/
  ↪ all_schema_versions/foo` and VALUE is `bar`.

- `tidb-ctl etcd delkey` deletes the KEY in etcd. Only those KEYs with the `/tidb/
  ↪ ddl/fg/owner/` or `/tidb/ddl/all_schema_versions/` prefix can be deleted.

```
tidb-ctl etcd delkey "/tidb/ddl/fg/owner/foo"
tidb-ctl etcd delkey "/tidb/ddl/all_schema_versions/bar"
```

4.16.14.2.6 The `log` command

The stack information for the TiDB error log is in one line format. You could use `tidb-ctl log` to change its format to multiple lines.

4.16.14.2.7 The `keyrange` command

The `keyrange` subcommand is used to query the global or table-related key range information, which is output in the hexadecimal form.

- Execute the `tidb-ctl keyrange` command to check the global key range information:

```
tidb-ctl keyrange
```

```
global ranges:
  meta: (6d, 6e)
  table: (74, 75)
```

- Add the `--encode` option to display encoded keys (in the same format as in TiKV and PD):

```
tidb-ctl keyrange --encode
```

```
global ranges:
  meta: (6d00000000000000f8, 6e00000000000000f8)
  table: (7400000000000000f8, 7500000000000000f8)
```

- Execute the `tidb-ctl keyrange --database={db} --table={tbl}` command to check the global and table-related key range information:

```
tidb-ctl keyrange --database test --table ttt
```

```
global ranges:
  meta: (6d, 6e)
  table: (74, 75)
table ttt ranges: (NOTE: key range might be changed after DDL)
  table: (748000000000002f, 748000000000000030)
  table indexes: (74800000000000002f5f69, 74800000000000002f5f72)
    index c2: (74800000000000002f5f698000000000000001,
       ↪ 74800000000000002f5f698000000000000002)
    index c3: (74800000000000002f5f698000000000000002,
       ↪ 74800000000000002f5f698000000000000003)
    index c4: (74800000000000002f5f698000000000000003,
       ↪ 74800000000000002f5f698000000000000004)
  table rows: (74800000000000002f5f72, 748000000000000030)
```

5 Deploy a TiDB Cluster in Kubernetes

You can use TiDB Operator to deploy TiDB clusters in Kubernetes. TiDB Operator is an automatic operation system for TiDB clusters in Kubernetes. It provides full life-cycle management for TiDB including deployment, upgrades, scaling, backup, fail-over, and configuration changes. With TiDB Operator, TiDB can run seamlessly in the Kubernetes clusters deployed on a public or private cloud.

Currently, the TiDB in Kubernetes documentation is independent of the TiDB documentation. For detailed steps on how to deploy TiDB clusters in Kubernetes using TiDB Operator, see TiDB in Kubernetes documentation.

6 FAQs

6.1 TiDB FAQ

This document lists the Most Frequently Asked Questions about TiDB.

6.1.1 About TiDB

6.1.1.1 TiDB introduction and architecture

6.1.1.1.1 What is TiDB?

TiDB is a distributed SQL database that features in horizontal scalability, high availability and consistent distributed transactions. It also enables you to use MySQL's SQL syntax and protocol to manage and retrieve data.

6.1.1.1.2 What is TiDB's architecture?

The TiDB cluster has three components: the TiDB server, the PD (Placement Driver) server, and the TiKV server. For more details, see TiDB architecture.

6.1.1.1.3 Is TiDB based on MySQL?

No. TiDB supports MySQL syntax and protocol, but it is a new open source database that is developed and maintained by PingCAP, Inc.

6.1.1.1.4 What is the respective responsibility of TiDB, TiKV and PD (Placement Driver)?

- TiDB works as the SQL computing layer, mainly responsible for parsing SQL, specifying query plan, and generating executor.
- TiKV works as a distributed Key-Value storage engine, used to store the real data. In short, TiKV is the storage engine of TiDB.
- PD works as the cluster manager of TiDB, which manages TiKV metadata, allocates timestamps, and makes decisions for data placement and load balancing.

6.1.1.1.5 Is it easy to use TiDB?

Yes, it is. When all the required services are started, you can use TiDB as easily as a MySQL server. You can replace MySQL with TiDB to power your applications without changing a single line of code in most cases. You can also manage TiDB using the popular MySQL management tools.

6.1.1.1.6 How is TiDB compatible with MySQL?

Currently, TiDB supports the majority of MySQL 5.7 syntax, but does not support trigger, stored procedures, user-defined functions, and foreign keys. For more details, see Compatibility with MySQL.

If you use the MySQL 8.0 client and it fails to connect to TiDB, try to add the `default-auth` and `default-character-set` options:

```
mysql -h 127.0.0.1 -u root -P 4000 --default-auth=mysql_native_password --default-character-set=utf8
```

This problem occurs because MySQL 8.0 changes the authentication plugin default in MySQL 5.7. To solve this problem, you need to add the options above to specify using the old encryption method.

6.1.1.1.7 How is TiDB highly available?

TiDB is self-healing. All of the three components, TiDB, TiKV and PD, can tolerate failures of some of their instances. With its strong consistency guarantee, whether it's data machine failures or even downtime of an entire data center, your data can be recovered automatically. For more information, see TiDB architecture.

6.1.1.1.8 How is TiDB strongly consistent?

TiDB implements Snapshot Isolation consistency, which it advertises as `REPEATABLE-READ` for compatibility with MySQL. Data is redundantly copied between TiKV nodes using the Raft consensus algorithm to ensure recoverability should a node failure occur.

At the bottom layer, TiKV uses a model of replication log + State Machine to replicate data. For the write requests, the data is written to a Leader and the Leader then replicates the command to its Followers in the form of log. When the majority of nodes in the cluster receive this log, this log is committed and can be applied into the State Machine.

6.1.1.1.9 Does TiDB support distributed transactions?

Yes. TiDB distributes transactions across your cluster, whether it is a few nodes in a single location or many nodes across multiple datacenters.

Inspired by Google's Percolator, the transaction model in TiDB is mainly a two-phase commit protocol with some practical optimizations. This model relies on a timestamp allocator to assign the monotone increasing timestamp for each transaction, so conflicts can be detected. PD works as the timestamp allocator in a TiDB cluster.

6.1.1.1.10 What programming language can I use to work with TiDB?

Any language supported by MySQL client or driver.

6.1.1.1.11 Can I use other Key-Value storage engines with TiDB?

Yes. TiKV and TiDB support many popular standalone storage engines, such as GolevelDB and BoltDB. If the storage engine is a KV engine that supports transactions and it provides a client that meets the interface requirement of TiDB, then it can connect to TiDB.

6.1.1.1.12 What's the recommended solution for the deployment of three geo-distributed data centers?

The architecture of TiDB guarantees that it fully supports geo-distribution and multi-activeness. Your data and applications are always-on. All the outages are transparent to your applications and your data can recover automatically. The operation depends on the network latency and stability. It is recommended to keep the latency within 5ms. Currently, we already have similar use cases. For details, contact info@pingcap.com.

6.1.1.1.13 Does TiDB provide any other knowledge resource besides the documentation?

Currently, TiDB documentation is the most important and timely way to get knowledge of TiDB. In addition, we also have some technical communication groups. If you have any needs, contact info@pingcap.com.

6.1.1.1.14 What are the MySQL variables that TiDB is compatible with?

See The System Variables.

6.1.1.1.15 The order of results is different from MySQL when `ORDER BY` is omitted

It is not a bug. The default order of records depends on various situations without any guarantee of consistency.

The order of results in MySQL might appear stable because queries are executed in a single thread. However, it is common that query plans can change when upgrading to new versions. It is recommended to use `ORDER BY` whenever an order of results is desired.

The reference can be found in ISO/IEC 9075:1992, Database Language SQL- July 30, 1992, which states as follows:

> If an `<order by clause>` is not specified, then the table specified by the `<cursor specification>` is T and the ordering of rows in T is implementation-dependent. In the following two queries, both results are considered legal:

```
> select * from t;
+------+------+
| a    | b    |
+------+------+
|    1 |    1 |
|    2 |    2 |
+------+------+
2 rows in set (0.00 sec)
```

```
> select * from t; -- the order of results is not guaranteed
+------+------+
| a    | b    |
+------+------+
|    2 |    2 |
|    1 |    1 |
+------+------+
2 rows in set (0.00 sec)
```

A statement is also considered non-deterministic if the list of columns used in the `ORDER BY` is non-unique. In the following example, the column `a` has duplicate values. Thus, only `ORDER BY a, b` would be guaranteed deterministic:

```
> select * from t order by a;
+------+------+
| a    | b    |
+------+------+
|    1 |    1 |
|    2 |    1 |
|    2 |    2 |
+------+------+
3 rows in set (0.00 sec)
```

```
> select * from t order by a; -- the order of column a is guaranteed, but
    ↪ b is not
+------+------+
| a    | b    |
+------+------+
|    1 |    1 |
|    2 |    2 |
|    2 |    1 |
+------+------+
3 rows in set (0.00 sec)
```

6.1.1.1.16 Does TiDB support `SELECT FOR UPDATE`?

Yes. When using pessimistic locking (the default since TiDB v3.0) the `SELECT FOR UPDATE` execution behaves similar to MySQL.

When using optimistic locking, `SELECT FOR UPDATE` does not lock data when the transaction is started, but checks conflicts when the transaction is committed. If the check reveals conflicts, the committing transaction rolls back.

6.1.1.1.17 Can the codec of TiDB guarantee that the UTF-8 string is memcomparable? Is there any coding suggestion if our key needs to support UTF-8?

The character sets of TiDB use UTF-8 by default and currently only support UTF-8. The string of TiDB uses the memcomparable format.

6.1.1.1.18 What is the length limit for the TiDB user name?

32 characters at most.

6.1.1.1.19 What is the maximum number of statements in a transaction?

5000 at most.

6.1.1.1.20 Does TiDB support XA?

No. The JDBC driver of TiDB is MySQL JDBC (Connector/J). When using Atomikos, set the data source to `type="com.mysql.jdbc.jdbc2.optional.MysqlXADataSource"`. TiDB does not support the connection with MySQL JDBC XADataSource. MySQL JDBC XADataSource only works for MySQL (for example, using DML to modify the `redo` log).

After you configure the two data sources of Atomikos, set the JDBC drives to XA. When Atomikos operates TM and RM (DB), Atomikos sends the command including XA to the JDBC layer. Taking MySQL for an example, when XA is enabled in the JDBC layer, JDBC will send a series of XA logic operations to InnoDB, including using DML to change the `redo` log. This is the operation of the two-phase commit. The current TiDB version does not support the upper application layer JTA/XA and does not parse XA operations sent by Atomikos.

As a standalone database, MySQL can only implement across-database transactions using XA; while TiDB supports distributed transactions using Google Percolator transaction model and its performance stability is higher than XA, so TiDB does not support XA and there is no need for TiDB to support XA.

6.1.1.1.21 Does `show processlist` display the system process ID?

The display content of TiDB `show processlist` is almost the same as that of MySQL `show processlist`. TiDB `show processlist` does not display the system process ID.

The ID that it displays is the current session ID. The differences between TiDB `show processlist` and MySQL `show processlist` are as follows:

- As TiDB is a distributed database, the `tidb-server` instance is a stateless engine for parsing and executing the SQL statements (for details, see TiDB architecture). `show processlist` displays the session list executed in the `tidb-server` instance that the user logs in to from the MySQL client, not the list of all the sessions running in the cluster. But MySQL is a standalone database and its `show processlist` displays all the SQL statements executed in MySQL.
- The `State` column in TiDB is not continually updated during query execution. As TiDB supports parallel query, each statement may be in multiple *states* at once, and thus it is difficult to simplify to a single value.

6.1.1.1.22 How to modify the user password and privilege?

To modify the user password in TiDB, it is recommended to use `set password for 'root'@'%' = '0101001';` or `alter`, not `update mysql.user` which might lead to the condition that the password in other nodes is not refreshed timely.

It is recommended to use the official standard statements when modifying the user password and privilege. For details, see TiDB user account management.

6.1.1.1.23 Why does the auto-increment ID of the later inserted data is smaller than that of the earlier inserted data in TiDB?

The auto-increment ID feature in TiDB is only guaranteed to be automatically incremental and unique but is not guaranteed to be allocated sequentially. Currently, TiDB is allocating IDs in batches. If data is inserted into multiple TiDB servers simultaneously, the allocated IDs are not sequential. When multiple threads concurrently insert data to multiple `tidb-server` instances, the auto-increment ID of the later inserted data may be smaller. TiDB allows specifying `AUTO_INCREMENT` for the integer field, but allows only one `AUTO_INCREMENT` field in a single table. For details, see MySQL Compatibility.

6.1.1.1.24 How do I modify the `sql_mode` in TiDB?

TiDB supports modifying the `sql_mode` as a system variable, as in MySQL. Currently, TiDB does not permit modifying the sql mode in a configuration file, but system variable changes made with `SET GLOBAL` propagate to all TiDB servers in the cluster and persist across restarts.

6.1.1.1.25 Does TiDB support modifying the MySQL version string of the server to a specific one that is required by the security vulnerability scanning tool?

Since v3.0.8, TiDB supports modifying the version string of the server by modifying `server-version` in the configuration file. When you deploy TiDB using TiDB Ansible, you

can also specify the proper version string by configuring `server-version` in the `conf/tidb .yml` configuration file to avoid the failure of security vulnerability scan.

6.1.1.1.26 What authentication protocols does TiDB support? What's the process?

- Like MySQL, TiDB supports the SASL protocol for user login authentication and password processing.
- When the client connects to TiDB, the challenge-response authentication mode starts. The process is as follows:

    1. The client connects to the server.
    2. The server sends a random string challenge to the client.
    3. The client sends the username and response to the server.
    4. The server verifies the response.

6.1.1.2 TiDB techniques

6.1.1.2.1 TiKV for data storage

See TiDB Internal (I) - Data Storage.

6.1.1.2.2 TiDB for data computing

See TiDB Internal (II) - Computing.

6.1.1.2.3 PD for scheduling

See TiDB Internal (III) - Scheduling.

6.1.2 Install, deploy and upgrade

6.1.2.1 Prepare

6.1.2.1.1 Operating system version requirements

Linux OS Platform	Version
Red Hat Enterprise Linux	7.3 or later
CentOS	7.3 or later
Oracle Enterprise Linux	7.3 or later

Why it is recommended to deploy the TiDB cluster on CentOS 7?

As an open source distributed NewSQL database with high performance, TiDB can be deployed in the Intel architecture server and major virtualization environments and runs well. TiDB supports most of the major hardware networks and Linux operating systems. For details, see Software and Hardware Requirements for deploying TiDB.

6.1.2.1.2 Server requirements

You can deploy and run TiDB on the 64-bit generic hardware server platform in the Intel x86-64 architecture. The requirements and recommendations about server hardware configuration for development, testing and production environments are as follows:

Development and testing environments

Component	CPU	Memory	Local Storage	Network	Instance Number (Minimum Requirement)
TiDB	8 core+	16 GB+	SAS, 200 GB+	Gigabit network card	1 (can be deployed on the same machine with PD)
PD	8 core+	16 GB+	SAS, 200 GB+	Gigabit network card	1 (can be deployed on the same machine with TiDB)
TiKV	8 core+	32 GB+	SAS, 200 GB+	Gigabit network card	3
				Total Server Number	4

Production environment

Component	CPU	Memory	Hard Disk Type	Network	Instance Nu
TiDB	16 core+	48 GB+	SAS	10 Gigabit network card (2 preferred)	
PD	8 core+	16 GB+	SSD	10 Gigabit network card (2 preferred)	
TiKV	16 core+	48 GB+	SSD	10 Gigabit network card (2 preferred)	
Monitor	8 core+	16 GB+	SAS	Gigabit network card	
				Total Server Number	

What's the purposes of 2 network cards of 10 gigabit?

As a distributed cluster, TiDB has a high demand on time, especially for PD, because PD needs to distribute unique timestamps. If the time in the PD servers is not consistent, it takes

longer waiting time when switching the PD server. The bond of two network cards guarantees the stability of data transmission, and 10 gigabit guarantees the transmission speed. Gigabit network cards are prone to meet bottlenecks, therefore it is strongly recommended to use 10 gigabit network cards.

Is it feasible if we don't use RAID for SSD?

If the resources are adequate, it is recommended to use RAID 10 for SSD. If the resources are inadequate, it is acceptable not to use RAID for SSD.

What's the recommended configuration of TiDB components?

- TiDB has a high requirement on CPU and memory. If you need to open Binlog, the local disk space should be increased based on the service volume estimation and the time requirement for the GC operation. But the SSD disk is not a must.
- PD stores the cluster metadata and has frequent Read and Write requests. It demands a high I/O disk. A disk of low performance will affect the performance of the whole cluster. It is recommended to use SSD disks. In addition, a larger number of Regions has a higher requirement on CPU and memory.
- TiKV has a high requirement on CPU, memory and disk. It is required to use SSD.

For details, see Software and Hardware Recommendations.

6.1.2.2 Install and deploy

6.1.2.2.1 Deploy TiDB using TiDB Ansible (recommended)

See Ansible Deployment.

Why the modified `toml` configuration for TiKV/PD does not take effect?

You need to set the `--config` parameter in TiKV/PD to make the `toml` configuration effective. TiKV/PD does not read the configuration by default. Currently, this issue only occurs when deploying using Binary. For TiKV, edit the configuration and restart the service. For PD, the configuration file is only read when PD is started for the first time, after which you can modify the configuration using pd-ctl. For details, see PD Control User Guide.

Should I deploy the TiDB monitoring framework (Prometheus + Grafana) on a standalone machine or on multiple machines? What is the recommended CPU and memory?

The monitoring machine is recommended to use standalone deployment. It is recommended to use an 8 core CPU with 16 GB+ memory and a 500 GB+ hard disk.

Why the monitor cannot display all metrics?

Check the time difference between the machine time of the monitor and the time within the cluster. If it is large, you can correct the time and the monitor will display all the metrics.

What is the function of supervise/svc/svstat service?

- supervise: the daemon process, to manage the processes
- svc: to start and stop the service
- svstat: to check the process status

Description of inventory.ini variables

Variable	Description
cluster_name	the name of a cluster, adjustable
tidb_version	the version of TiDB, configured by default in TiDB Ansible branches
deployment_method	the method of deployment, binary by default, Docker optional

Variable	Description
process_supervision	the supervision way of processes, systemd by default, supervise optional
timezone	the timezone of the managed node, adjustable, `Asia/Shanghai` by default, used with the `set_timezone` variable
set_timezone	to edit the timezone of the managed node, True by default; False means closing

Variable	Description
enable_ntpd	currently not supported
enable_firewalld	to enable the firewall, closed by default
enable_ntpd	to monitor the NTP service of the managed node, True by default; do not close it
machine_benchmark	to monitor the disk IOPS of the managed node, True by default; do not close it

Variable	Description
set_hostname	to edit the host-name of the man-aged node based on the IP, False by default
enable_binlog	whether to deploy Pump and enable the binlog, False by default, depen-dent on the Kafka cluster; see the `zookeeper_addrs` ↪ variable
zookeeper_addrs	the ZooKeeper address of the binlog Kafka cluster

Variable	Description
enable_slow_query_log	to record the slow query log of TiDB into a single file: ({{ deploy_dir }}/log/tidb_slow_query.log). False by default, to record it into the TiDB log

Variable	Description
deploy_without_tidb	Key-Value mode, deploy only PD, TiKV and the monitoring service, not TiDB; set the IP of the tidb_servers host group to null in the `inventory.ini` file

6.1.2.2.2 Deploy TiDB offline using TiDB Ansible

It is not recommended to deploy TiDB offline using TiDB Ansible. If the Control Machine has no access to external network, you can deploy TiDB offline using TiDB Ansible. For details, see Offline Deployment Using TiDB Ansible.

6.1.2.2.3 How to deploy TiDB quickly using Docker Compose on a single machine?

You can use Docker Compose to build a TiDB cluster locally, including the cluster monitoring components. You can also customize the version and number of instances for each component. The configuration file can also be customized. You can only use this deployment method for testing and development environment. For details, see Building the Cluster Using Docker Compose.

6.1.2.2.4 How to separately record the slow query log in TiDB? How to locate the slow query SQL statement?

1. The slow query definition for TiDB is in the `conf/tidb.yml` configuration file of `tidb-ansible`. The `slow-threshold: 300` parameter is used to configure the threshold value of the slow query (unit: millisecond).

   The slow query log is recorded in `tidb.log` by default. If you want to generate a slow query log file separately, set `enable_slow_query_log` in the `inventory.ini` configuration file to `True`.

   Then run `ansible-playbook rolling_update.yml --tags=tidb` to perform a rolling update on the `tidb-server` instance. After the update is finished, the `tidb-server` instance will record the slow query log in `tidb_slow_query.log`.

2. If a slow query occurs, you can locate the `tidb-server` instance where the slow query is and the slow query time point using Grafana and find the SQL statement information recorded in the log on the corresponding node.

3. In addition to the log, you can also view the slow query using the `admin show slow` command. For details, see `admin show slow` command.

6.1.2.2.5 How to add the `label` configuration if `label` of TiKV was not configured when I deployed the TiDB cluster for the first time?

The configuration of TiDB `label` is related to the cluster deployment architecture. It is important and is the basis for PD to execute global management and scheduling. If you did not configure `label` when deploying the cluster previously, you should adjust the deployment structure by manually adding the `location-labels` information using the PD management tool `pd-ctl`, for example, `config set location-labels "zone,rack,host"` (you should configure it based on the practical `label` level name).

For the usage of `pd-ctl`, see PD Control Instruction.

6.1.2.2.6 Why does the `dd` command for the disk test use the `oflag=direct` option?

The Direct mode wraps the Write request into the I/O command and sends this command to the disk to bypass the file system cache and directly test the real I/O Read/Write performance of the disk.

6.1.2.2.7 How to use the `fio` command to test the disk performance of the TiKV instance?

- Random Read test:

```
./fio -ioengine=psync -bs=32k -fdatasync=1 -thread -rw=randread -size
    ↪ =10G -filename=fio_randread_test.txt -name='fio randread test' -
    ↪ iodepth=4 -runtime=60 -numjobs=4 -group_reporting --output-format
    ↪ =json --output=fio_randread_result.json
```

- The mix test of sequential Write and random Read:

```
./fio -ioengine=psync -bs=32k -fdatasync=1 -thread -rw=randrw -
    ↪ percentage_random=100,0 -size=10G -filename=
    ↪ fio_randread_write_test.txt -name='fio mixed randread and
    ↪ sequential write test' -iodepth=4 -runtime=60 -numjobs=4 -
    ↪ group_reporting --output-format=json --output=
    ↪ fio_randread_write_test.json
```

6.1.2.2.8 Error `UNREACHABLE! "msg": "Failed to connect to the host via ssh: "` when deploying TiDB using TiDB Ansible

Two possible reasons and solutions:

- The SSH mutual trust is not configured as required. It's recommended to follow the steps described in the official document and check whether it is successfully configured using `ansible -i inventory.ini all -m shell -a 'whoami' -b`.
- If it involves the scenario where a single server is assigned multiple roles, for example, the mixed deployment of multiple components or multiple TiKV instances are deployed on a single server, this error might be caused by the SSH reuse mechanism. You can use the option of `ansible ⋯ -f 1` to avoid this error.

6.1.2.3 Upgrade

6.1.2.3.1 How to perform rolling updates using TiDB Ansible?

- Apply rolling updates to the TiKV node (only update the TiKV service).

```
ansible-playbook rolling_update.yml --tags=tikv
```

- Apply rolling updates to all services.

```
ansible-playbook rolling_update.yml
```

6.1.2.3.2 How are the rolling updates done?

When you apply rolling updates to the TiDB services, the running application is not affected. You need to configure the minimum cluster topology (TiDB * 2, PD * 3, TiKV * 3). If the Pump/Drainer service is involved in the cluster, it is recommended to stop Drainer before rolling updates. When you update TiDB, Pump is also updated.

6.1.2.3.3 How to upgrade when I deploy TiDB using Binary?

It is not recommended to deploy TiDB using Binary. The support for upgrading using Binary is not as friendly as using TiDB Ansible. It is recommended to deploy TiDB using TiDB Ansible.

6.1.2.3.4 Should I upgrade TiKV or all components generally?

Generally you should upgrade all components, because the whole version is tested together. Upgrade a single component only when an emergent issue occurs and you need to upgrade this component.

6.1.2.3.5 What causes "Timeout when waiting for search string 200 OK" when starting or upgrading a cluster? How to deal with it?

Possible reasons:

- The process did not start normally.
- The port is occupied.
- The process did not stop normally.
- You use `rolling_update.yml` to upgrade the cluster when the cluster is stopped (operation error).

Solution:

- Log into the node to check the status of the process or port.
- Correct the incorrect operation procedure.

6.1.3 Manage the cluster

6.1.3.1 Daily management

6.1.3.1.1 What are the common operations?

Job	Playbook
Start the cluster	`ansible-playbook start.yml`
Stop the cluster	`ansible-playbook stop.yml`
Destroy the cluster	`ansible-playbook unsafe_cleanup.yml` (If the deployment directory is a mount point, an error will be reported, but implementation results will remain unaffected)
Clean data (for test)	`ansible-playbook unsafe_cleanup_data.yml`
Apply rolling updates	`ansible-playbook rolling_update.yml`

Job	Playbook
Apply rolling updates to TiKV	`ansible-playbook rolling_update.yml --tags=tikv`
Apply rolling updates to components except PD	`ansible-playbook rolling_update.yml --skip-tags=pd`
Apply rolling updates to the monitoring components	`ansible-playbook rolling_update_monitor.yml`

6.1.3.1.2 How to log into TiDB?

You can log into TiDB like logging into MySQL. For example:

```
mysql -h 127.0.0.1 -uroot -P4000
```

6.1.3.1.3 How to modify the system variables in TiDB?

Similar to MySQL, TiDB includes static and solid parameters. You can directly modify static parameters using `set global xxx = n`, but the new value of a parameter is only effective within the life cycle in this instance.

6.1.3.1.4 Where and what are the data directories in TiDB (TiKV)?

TiKV data is located in the `--data-dir`, which include four directories of backup, db, raft, and snap, used to store backup, data, Raft data, and mirror data respectively.

6.1.3.1.5 What are the system tables in TiDB?

Similar to MySQL, TiDB includes system tables as well, used to store the information required by the server when it runs.

6.1.3.1.6 Where are the TiDB/PD/TiKV logs?

By default, TiDB/PD/TiKV outputs standard error in the logs. If a log file is specified by `--log-file` during the startup, the log is output to the specified file and executes rotation daily.

6.1.3.1.7 How to safely stop TiDB?

If the cluster is deployed using TiDB Ansible, you can use the `ansible-playbook stop.yml` command to stop the TiDB cluster. If the cluster is not deployed using TiDB Ansible, `kill` all the services directly. The components of TiDB will do `graceful shutdown`.

6.1.3.1.8 Can `kill` be executed in TiDB?

- You can `kill` DML statements. First use `show processlist` to find the ID corresponding with the session, and then run `kill tidb [session id]`.
- You can `kill` DDL statements. First use `admin show ddl jobs` to find the ID of the DDL job you need to kill, and then run `admin cancel ddl jobs 'job_id' [, 'job_id'] ...`. For more details, see the ADMIN statement.

6.1.3.1.9 Does TiDB support session timeout?

TiDB supports the MySQL configuration settings `wait_timeout` and `interactive_timeout`. By configuring these settings, it is possible to have sessions automatically timeout after an idle period.

6.1.3.1.10 What is the TiDB version management strategy for production environment? How to avoid frequent upgrade?

Currently, TiDB has a standard management of various versions. Each release contains a detailed change log and release notes. Whether it is necessary to upgrade in the production environment depends on the application system. It is recommended to learn the details about the functional differences between the previous and later versions before upgrading.

Take `Release Version: v1.0.3-1-ga80e796` as an example of version number description:

- `v1.0.3` indicates the standard GA version.
- `-1` indicates the current version has one commit.
- `ga80e796` indicates the version `git-hash`.

6.1.3.1.11 What's the difference between various TiDB master versions? How to avoid using the wrong TiDB Ansible version?

The TiDB community is highly active. After the 1.0 GA release, the engineers have been keeping optimizing and fixing bugs. Therefore, the TiDB version is updated quite fast. If you want to keep informed of the latest version, see TiDB Weekly update.

It is recommended to deploy the TiDB cluster using the latest version of TiDB Ansible, which will also be updated along with the TiDB version. TiDB has a unified management of the version number after the 1.0 GA release. You can view the version number using the following two methods:

- `select tidb_version()`
- `tidb-server -V`

6.1.3.1.12 Is there a graphical deployment tool for TiDB?

Currently no.

6.1.3.1.13 How to scale TiDB horizontally?

As your business grows, your database might face the following three bottlenecks:

- Lack of storage resources which means that the disk space is not enough.
- Lack of computing resources such as high CPU occupancy.
- Not enough write and read capacity.

You can scale TiDB as your business grows.

- If the disk space is not enough, you can increase the capacity simply by adding more TiKV nodes. When the new node is started, PD will migrate the data from other nodes to the new node automatically.
- If the computing resources are not enough, check the CPU consumption situation first before adding more TiDB nodes or TiKV nodes. When a TiDB node is added, you can configure it in the Load Balancer.
- If the capacity is not enough, you can add both TiDB nodes and TiKV nodes.

6.1.3.1.14 Why does TiDB use gRPC instead of Thrift? Is it because Google uses it?

Not really. We need some good features of gRPC, such as flow control, encryption and streaming.

6.1.3.1.15 What does the 92 indicate in `like(bindo.customers.name, jason%, 92)`?

The 92 indicates the escape character, which is ASCII 92 by default.

6.1.3.1.16 Why does the data length shown by `information_schema.tables.data_length` differ from the store size on the TiKV monitoring panel?

Two reasons:

- The two results are calculated in different ways. `information_schema.tables.` ↪ `data_length` is an estimated value by calculating the averaged length of each row, while the store size on the TiKV monitoring panel sums up the length of the data files (the SST files of RocksDB) in a single TiKV instance.
- `information_schema.tables.data_length` is a logical value, while the store size is a physical value. The redundant data generated by multiple versions of the transaction is not included in the logical value, while the redundant data is compressed by TiKV in the physical value.

6.1.3.2 Manage the PD server

6.1.3.2.1 The `TiKV cluster is not bootstrapped` message is displayed when I access PD

Most of the APIs of PD are available only when the TiKV cluster is initialized. This message is displayed if PD is accessed when PD is started while TiKV is not started when a new cluster is deployed. If this message is displayed, start the TiKV cluster. When TiKV is initialized, PD is accessible.

6.1.3.2.2 The `etcd cluster ID mismatch` message is displayed when starting PD

This is because the `--initial-cluster` in the PD startup parameter contains a member that doesn't belong to this cluster. To solve this problem, check the corresponding cluster of each member, remove the wrong member, and then restart PD.

6.1.3.2.3 What's the maximum tolerance for time synchronization error of PD?

PD can tolerate any synchronization error, but a larger error value means a larger gap between the timestamp allocated by the PD and the physical time, which will affect functions such as read of historical versions.

6.1.3.2.4 How does the client connection find PD?

The client connection can only access the cluster through TiDB. TiDB connects PD and TiKV. PD and TiKV are transparent to the client. When TiDB connects to any PD, the PD tells TiDB who is the current leader. If this PD is not the leader, TiDB reconnects to the leader PD.

6.1.3.2.5 What is the difference between the `leader-schedule-limit` and `region-schedule-limit` scheduling parameters in PD?

- The `leader-schedule-limit` scheduling parameter is used to balance the Leader number of different TiKV servers, affecting the load of query processing.
- The `region-schedule-limit` scheduling parameter is used to balance the replica number of different TiKV servers, affecting the data amount of different nodes.

6.1.3.2.6 Is the number of replicas in each region configurable? If yes, how to configure it?

Yes. Currently, you can only update the global number of replicas. When started for the first time, PD reads the configuration file (conf/pd.yml) and uses the max-replicas configuration in it. If you want to update the number later, use the pd-ctl configuration command

`config set max-replicas $num` and view the enabled configuration using `config show all`. The updating does not affect the applications and is configured in the background.

Make sure that the total number of TiKV instances is always greater than or equal to the number of replicas you set. For example, 3 replicas need 3 TiKV instances at least. Additional storage requirements need to be estimated before increasing the number of replicas. For more information about pd-ctl, see PD Control User Guide.

6.1.3.2.7 How to check the health status of the whole cluster when lacking command line cluster management tools?

You can determine the general status of the cluster using the pd-ctl tool. For detailed cluster status, you need to use the monitor to determine.

6.1.3.2.8 How to delete the monitoring data of a cluster node that is offline?

The offline node usually indicates the TiKV node. You can determine whether the offline process is finished by the pd-ctl or the monitor. After the node is offline, perform the following steps:

1. Manually stop the relevant services on the offline node.
2. Delete the `node_exporter` data of the corresponding node from the Prometheus configuration file.
3. Delete the data of the corresponding node from Ansible `inventory.ini`.

6.1.3.2.9 Why couldn't I connect to the PD server using `127.0.0.1` when I was using the PD Control?

If your TiDB cluster is deployed using TiDB Ansible, the PD external service port is not bound to `127.0.0.1`, so PD Control does not recognize `127.0.0.1` and you can only connect to it using the local IP address.

6.1.3.3 Manage the TiDB server

6.1.3.3.1 How to set the `lease` parameter in TiDB?

The lease parameter (`--lease=60`) is set from the command line when starting a TiDB server. The value of the lease parameter impacts the Database Schema Changes (DDL) speed of the current session. In the testing environments, you can set the value to 1s for to speed up the testing cycle. But in the production environments, it is recommended to set the value to minutes (for example, 60) to ensure the DDL safety.

6.1.3.3.2 What is the processing time of a DDL operation?

The processing time is different for different scenarios. Generally, you can consider the following three scenarios:

1. The `Add Index` operation with a relatively small number of rows in the corresponding data table: about 3s
2. The `Add Index` operation with a relatively large number of rows in the corresponding data table: the processing time depends on the specific number of rows and the QPS at that time (the `Add Index` operation has a lower priority than ordinary SQL operations)
3. Other DDL operations: about 1s

If the TiDB server instance that receives the DDL request is the same TiDB server instance that the DDL owner is in, the first and third scenarios above may cost only dozens to hundreds of milliseconds.

6.1.3.3.3 Why it is very slow to run DDL statements sometimes?

Possible reasons:

- If you run multiple DDL statements together, the last few DDL statements might run slowly. This is because the DDL statements are executed serially in the TiDB cluster.
- After you start the cluster successfully, the first DDL operation may take a longer time to run, usually around 30s. This is because the TiDB cluster is electing the leader that processes DDL statements.
- The processing time of DDL statements in the first ten minutes after starting TiDB would be much longer than the normal case if you meet the following conditions: 1) TiDB cannot communicate with PD as usual when you are stopping TiDB (including the case of power failure); 2) TiDB fails to clean up the registration data from PD in time because TiDB is stopped by the `kill -9` command. If you run DDL statements during this period, for the state change of each DDL, you need to wait for 2 * lease (lease = 45s).
- If a communication issue occurs between a TiDB server and a PD server in the cluster, the TiDB server cannot get or update the version information from the PD server in time. In this case, you need to wait for 2 * lease for the state processing of each DDL.

6.1.3.3.4 Can I use S3 as the backend storage engine in TiDB?

No. Currently, TiDB only supports the distributed storage engine and the Goleveldb/RocksDB/BoltDB engine.

6.1.3.3.5 Can the `Information_schema` support more real information?

As part of MySQL compatibility, TiDB supports a number of `INFORMATION_SCHEMA` ↪ tables. Many of these tables also have a corresponding SHOW command. For more information, see Information Schema.

6.1.3.3.6 What's the explanation of the TiDB Backoff type scenario?

In the communication process between the TiDB server and the TiKV server, the `Server is busy` or `backoff.maxsleep 20000ms` log message is displayed when processing a large volume of data. This is because the system is busy while the TiKV server processes data. At this time, usually you can view that the TiKV host resources usage rate is high. If this occurs, you can increase the server capacity according to the resources usage.

6.1.3.3.7 What's the maximum number of concurrent connections that TiDB supports?

By default, there is no limit on the maximum number of connections per TiDB server. A limit may be enforced by setting `max-server-connections` in the `config.toml` file. If too large concurrency leads to an increase of response time, it is recommended to increase the capacity by adding TiDB nodes.

6.1.3.3.8 How to view the creation time of a table?

The `create_time` of tables in the `information_schema` is the creation time.

6.1.3.3.9 What is the meaning of `EXPENSIVE_QUERY` in the TiDB log?

When TiDB is executing a SQL statement, the query will be `EXPENSIVE_QUERY` if each operator is estimated to process over 10000 pieces of data. You can modify the `tidb-server` configuration parameter to adjust the threshold and then restart the `tidb-server`.

6.1.3.3.10 How to control or change the execution priority of SQL commits?

TiDB supports changing the priority on a per-session, global or individual statement basis. Priority has the following meaning:

- `HIGH_PRIORITY`: this statement has a high priority, that is, TiDB gives priority to this statement and executes it first.
- `LOW_PRIORITY`: this statement has a low priority, that is, TiDB reduces the priority of this statement during the execution period.

You can combine the above two parameters with the DML of TiDB to use them. For example:

1. Adjust the priority by writing SQL statements in the database:

```
select HIGH_PRIORITY | LOW_PRIORITY count(*) from table_name;
insert HIGH_PRIORITY | LOW_PRIORITY into table_name insert_values;
delete HIGH_PRIORITY | LOW_PRIORITY from table_name;
```

```
update HIGH_PRIORITY | LOW_PRIORITY table_reference set assignment_list
    ↪ where where_condition;
replace HIGH_PRIORITY | LOW_PRIORITY into table_name;
```

2. The full table scan statement automatically adjusts itself to a low priority. `analyze` has a low priority by default.

6.1.3.3.11 What's the trigger strategy for `auto analyze` in TiDB?

Trigger strategy: `auto analyze` is automatically triggered when the number of pieces of data in a new table reaches 1000 and this table has no write operation within one minute.

When the modified number or the current total row number is larger than `tidb_auto_analyze_ratio`, the `analyze` statement is automatically triggered. The default value of `tidb_auto_analyze_ratio` is 0.5, indicating that this feature is enabled by default. To ensure safety, its minimum value is 0.3 when the feature is enabled, and it must be smaller than `pseudo-estimate-ratio` whose default value is 0.8, otherwise pseudo statistics will be used for a period of time. It is recommended to set `tidb_auto_analyze_ratio` to 0.5.

6.1.3.3.12 Can I use hints to override the optimizer behavior?

TiDB supports multiple hints to override the default query optimizer behavior. The basic usage is similar to MySQL, with several TiDB specific extensions:

```
SELECT column_name FROM table_name USE INDEX ( index_name ) WHERE
    ↪ where_condition;
```

6.1.3.3.13 Why the `Information schema is changed` error is reported?

TiDB handles the SQL statement using the `schema` of the time and supports online asynchronous DDL change. A DML statement and a DDL statement might be executed at the same time and you must ensure that each statement is executed using the same `schema`. Therefore, when the DML operation meets the ongoing DDL operation, the `Information ↪ schema is changed` error might be reported. Some improvements have been made to prevent too many error reportings during the DML operation.

Now, there are still a few reasons for this error reporting (the latter two are unrelated to tables):

- Some tables involved in the DML operation are the same tables involved in the ongoing DDL operation.
- The DML operation goes on for a long time. During this period, many DDL statements have been executed, which causes more than 1024 `schema` version changes. This value is set to `100` in TiDB before v3.0.5. Since v3.0.5, the default value is `1024` which can be modified by modifying the `tidb_max_delta_schema_count` variable.

- The TiDB server that accepts the DML request is not able to load `schema information` for a long time (possibly caused by the connection failure between TiDB and PD or TiKV). During this period, many DDL statements have been executed, which causes more than 100 `schema` version changes.

Note:

- Currently, TiDB does not cache all the `schema` version changes.
- For each DDL operation, the number of `schema` version changes is the same with the number of corresponding `schema state` version changes.
- Different DDL operations cause different number of `schema` version changes. For example, the `CREATE TABLE` statement causes one `schema` version change while the `ADD COLUMN` statement causes four.

6.1.3.3.14 What are the causes of the "Information schema is out of date" error

When executing a DML statement, if TiDB fails to load the latest schema within a DDL lease (45s by default), the `Information schema is out of date` error might occur. Possible causes are:

- The TiDB instance that executed this DML was killed, and the transaction execution corresponding to this DML statement took longer than a DDL lease. When the transaction was committed, the error occurred.
- TiDB failed to connect to PD or TiKV while executing this DML statement. As a result, TiDB failed to load schema within a DDL lease or disconnected from PD due to the keepalive setting.

6.1.3.3.15 Error is reported when executing DDL statements under high concurrency?

When you execute DDL statements (such as creating tables in batches) under high concurrency, a very few of these statements might fail because of key conflicts during the concurrent execution.

It is recommended to keep the number of concurrent DDL statements under 20. Otherwise, you need to retry the failed statements from the client.

6.1.3.4 Manage the TiKV server

6.1.3.4.1 What is the recommended number of replicas in the TiKV cluster? Is it better to keep the minimum number for high availability?

3 replicas for each Region is sufficient for a testing environment. However, you should never operate a TiKV cluster with under 3 nodes in a production scenario. Depending on infrastructure, workload, and resiliency needs, you may wish to increase this number.

6.1.3.4.2 The `cluster ID mismatch` message is displayed when starting TiKV

This is because the cluster ID stored in local TiKV is different from the cluster ID specified by PD. When a new PD cluster is deployed, PD generates random cluster IDs. TiKV gets the cluster ID from PD and stores the cluster ID locally when it is initialized. The next time when TiKV is started, it checks the local cluster ID with the cluster ID in PD. If the cluster IDs don't match, the `cluster ID mismatch` message is displayed and TiKV exits.

If you previously deploy a PD cluster, but then you remove the PD data and deploy a new PD cluster, this error occurs because TiKV uses the old data to connect to the new PD cluster.

6.1.3.4.3 The `duplicated store address` message is displayed when starting TiKV

This is because the address in the startup parameter has been registered in the PD cluster by other TiKVs. This error occurs when there is no data folder under the directory that TiKV `--store` specifies, but you use the previous parameter to restart the TiKV.

To solve this problem, use the `store delete` function to delete the previous store and then restart TiKV.

6.1.3.4.4 TiKV leader replicas and follower replicas use the same compression algorithm. Why the amount of disk space occupied is different?

TiKV stores data in the LSM tree, in which each layer has a different compression algorithm. If two replicas of the same data are located in different layers in two TiKV nodes, the two replicas might occupy different space.

6.1.3.4.5 What are the features of TiKV block cache?

TiKV implements the Column Family (CF) feature of RocksDB. By default, the KV data is eventually stored in the 3 CFs (default, write and lock) within RocksDB.

- The default CF stores real data and the corresponding parameter is in `[rocksdb.` ↪ `defaultcf]`. The write CF stores the data version information (MVCC) and index-related data, and the corresponding parameter is in `[rocksdb.writecf]`. The lock CF stores the lock information and the system uses the default parameter.

- The Raft RocksDB instance stores Raft logs. The default CF mainly stores Raft logs and the corresponding parameter is in `[raftdb.defaultcf]`.
- All CFs have a shared block-cache to cache data blocks and improve RocksDB read speed. The size of block-cache is controlled by the `block-cache-size` parameter. A larger value of the parameter means more hot data can be cached and is more favorable to read operation. At the same time, it consumes more system memory.
- Each CF has an individual write-buffer and the size is controlled by the `write-buffer-size` parameter.

6.1.3.4.6 Why it occurs that "TiKV channel full"?

- The Raftstore thread is too slow or blocked by I/O. You can view the CPU usage status of Raftstore.
- TiKV is too busy (CPU, disk I/O, etc.) and cannot manage to handle it.

6.1.3.4.7 Why does TiKV frequently switch Region leader?

- Leaders can not reach out to followers. E.g., network problem or node failure.
- Leader balance from PD. E.g., PD wants to transfer leaders from a hotspot node to others.

6.1.3.4.8 If a node is down, will the service be affected? If yes, how long?

TiKV uses Raft to replicate data among multiple replicas (by default 3 replicas for each Region). If one replica goes wrong, the other replicas can guarantee data safety. Based on the Raft protocol, if a single leader fails as the node goes down, a follower in another node is soon elected as the Region leader after a maximum of 2 * lease time (lease time is 10 seconds).

6.1.3.4.9 What are the TiKV scenarios that take up high I/O, memory, CPU, and exceed the parameter configuration?

Writing or reading a large volume of data in TiKV takes up high I/O, memory and CPU. Executing very complex queries costs a lot of memory and CPU resources, such as the scenario that generates large intermediate result sets.

6.1.3.4.10 Does TiKV support SAS/SATA disks or mixed deployment of SSD/SAS disks?

No. For OLTP scenarios, TiDB requires high I/O disks for data access and operation. As a distributed database with strong consistency, TiDB has some write amplification such as replica replication and bottom layer storage compaction. Therefore, it is recommended to use NVMe SSD as the storage disks in TiDB best practices. Mixed deployment of TiKV and PD is not supported.

6.1.3.4.11 Is the Range of the Key data table divided before data access?

No. It differs from the table splitting rules of MySQL. In TiKV, the table Range is dynamically split based on the size of Region.

6.1.3.4.12 How does Region split?

Region is not divided in advance, but it follows a Region split mechanism. When the Region size exceeds the value of the `region-max-size` or `region-max-keys` parameters, split is triggered. After the split, the information is reported to PD.

6.1.3.4.13 Does TiKV have the `innodb_flush_log_trx_commit` parameter like MySQL, to guarantee the security of data?

Yes. Currently, the standalone storage engine uses two RocksDB instances. One instance is used to store the raft-log. When the `sync-log` parameter in TiKV is set to true, each commit is mandatorily flushed to the raft-log. If a crash occurs, you can restore the KV data using the raft-log.

6.1.3.4.14 What is the recommended server configuration for WAL storage, such as SSD, RAID level, cache strategy of RAID card, NUMA configuration, file system, I/O scheduling strategy of the operating system?

WAL belongs to ordered writing, and currently, we do not apply a unique configuration to it. Recommended configuration is as follows:

- SSD
- RAID 10 preferred
- Cache strategy of RAID card and I/O scheduling strategy of the operating system: currently no specific best practices; you can use the default configuration in Linux 7 or later
- NUMA: no specific suggestion; for memory allocation strategy, you can use `interleave = all`
- File system: ext4

6.1.3.4.15 How is the write performance in the most strict data available mode (`sync-log = true`)?

Generally, enabling `sync-log` reduces about 30% of the performance. For write performance when `sync-log` is set to `false`, see Performance test result for TiDB using Sysbench.

6.1.3.4.16 Can Raft + multiple replicas in the TiKV architecture achieve absolute data safety? Is it necessary to apply the most strict mode (`sync-log = true`) to a standalone storage?

Data is redundantly replicated between TiKV nodes using the Raft consensus algorithm to ensure recoverability should a node failure occur. Only when the data has been written

into more than 50% of the replicas will the application return ACK (two out of three nodes). However, theoretically, two nodes might crash. Therefore, except for scenarios with less strict requirement on data safety but extreme requirement on performance, it is strongly recommended that you enable the `sync-log` mode.

As an alternative to using `sync-log`, you may also consider having five replicas instead of three in your Raft group. This would allow for the failure of two replicas, while still providing data safety.

For a standalone TiKV node, it is still recommended to enable the `sync-log` mode. Otherwise, the last write might be lost in case of a node failure.

6.1.3.4.17 Since TiKV uses the Raft protocol, multiple network roundtrips occur during data writing. What is the actual write delay?

Theoretically, TiDB has a write delay of 4 more network roundtrips than standalone databases.

6.1.3.4.18 Does TiDB have a InnoDB memcached plugin like MySQL which can directly use the KV interface and does not need the independent cache?

TiKV supports calling the interface separately. Theoretically, you can take an instance as the cache. Because TiDB is a distributed relational database, we do not support TiKV separately.

6.1.3.4.19 What is the Coprocessor component used for?

- Reduce the data transmission between TiDB and TiKV
- Make full use of the distributed computing resources of TiKV to execute computing pushdown

6.1.3.4.20 The error message `IO error: No space left on device While appending to file` is displayed

This is because the disk space is not enough. You need to add nodes or enlarge the disk space.

6.1.3.4.21 Why does the OOM (Out of Memory) error occur frequently in TiKV?

The memory usage of TiKV mainly comes from the block-cache of RocksDB, which is 40% of the system memory size by default. When the OOM error occurs frequently in TiKV, you should check whether the value of `block-cache-size` is set too high. In addition, when multiple TiKV instances are deployed on a single machine, you need to explicitly configure the parameter to prevent multiple instances from using too much system memory that results in the OOM error.

6.1.3.4.22 Can both TiDB data and RawKV data be stored in the same TiKV cluster?

No. TiDB (or data created from the transactional API) relies on a specific key format. It is not compatible with data created from RawKV API (or data from other RawKV-based services).

6.1.3.5 TiDB test

6.1.3.5.1 What is the performance test result for TiDB using Sysbench?

At the beginning, many users tend to do a benchmark test or a comparison test between TiDB and MySQL. We have also done a similar official test and find the test result is consistent at large, although the test data has some bias. Because the architecture of TiDB differs greatly from MySQL, it is hard to find a benchmark point. The suggestions are as follows:

- Do not spend too much time on the benchmark test. Pay more attention to the difference of scenarios using TiDB.
- See Performance test result for TiDB using Sysbench.

6.1.3.5.2 What's the relationship between the TiDB cluster capacity (QPS) and the number of nodes? How does TiDB compare to MySQL?

- Within 10 nodes, the relationship between TiDB write capacity (Insert TPS) and the number of nodes is roughly 40% linear increase. Because MySQL uses single-node write, its write capacity cannot be scaled.
- In MySQL, the read capacity can be increased by adding replicas, but the write capacity cannot be increased except using sharding, which has many problems.
- In TiDB, both the read and write capacity can be easily increased by adding more nodes.

6.1.3.5.3 The performance test of MySQL and TiDB by our DBA shows that the performance of a standalone TiDB is not as good as MySQL

TiDB is designed for scenarios where sharding is used because the capacity of a MySQL standalone is limited, and where strong consistency and complete distributed transactions are required. One of the advantages of TiDB is pushing down computing to the storage nodes to execute concurrent computing.

TiDB is not suitable for tables of small size (such as below ten million level), because its strength in concurrency cannot be shown with a small size of data and limited Regions. A typical example is the counter table, in which records of a few lines are updated high frequently. In TiDB, these lines become several Key-Value pairs in the storage engine, and then settle into a Region located on a single node. The overhead of background replication to guarantee strong consistency and operations from TiDB to TiKV leads to a poorer performance than a MySQL standalone.

6.1.3.6 Backup and restore

6.1.3.6.1 How to back up data in TiDB?

Currently, the preferred method for backup is using the PingCAP fork of Mydumper. Although the official MySQL tool `mysqldump` is also supported in TiDB to back up and restore data, its performance is poorer than `mydumper`/`loader` and it needs much more time to back up and restore large volumes of data.

Keep the size of the data file exported from `mydumper` as small as possible. It is recommended to keep the size within 64M. You can set value of the `-F` parameter to 64.

You can edit the `t` parameter of `loader` based on the number of TiKV instances and load status. For example, in scenarios of three TiKV instances, you can set its value to `3 * (1 ~ n)`. When the TiKV load is very high and `backoffer.maxSleep 15000ms is exceeded` displays a lot in `loader` and TiDB logs, you can adjust the parameter to a smaller value. When the TiKV load is not very high, you can adjust the parameter to a larger value accordingly.

6.1.4 Migrate the data and traffic

6.1.4.1 Full data export and import

6.1.4.1.1 Mydumper

See Mydumper Instructions.

6.1.4.1.2 Loader

See Loader Instructions.

6.1.4.1.3 How to migrate an application running on MySQL to TiDB?

Because TiDB supports most MySQL syntax, generally you can migrate your applications to TiDB without changing a single line of code in most cases.

6.1.4.1.4 If I accidentally import the MySQL user table into TiDB, or forget the password and cannot log in, how to deal with it?

Restart the TiDB service, add the `-skip-grant-table=true` parameter in the configuration file. Log into the cluster without password and recreate the user, or recreate the `mysql.user` table using the following statement:

```
DROP TABLE IF EXIST mysql.user;
```

```
CREATE TABLE if not exists mysql.user (
    Host      CHAR(64),
    User      CHAR(16),
    Password    CHAR(41),
    Select_priv   ENUM('N','Y') NOT NULL DEFAULT 'N',
    Insert_priv   ENUM('N','Y') NOT NULL DEFAULT 'N',
    Update_priv   ENUM('N','Y') NOT NULL DEFAULT 'N',
    Delete_priv   ENUM('N','Y') NOT NULL DEFAULT 'N',
    Create_priv   ENUM('N','Y') NOT NULL DEFAULT 'N',
    Drop_priv   ENUM('N','Y') NOT NULL DEFAULT 'N',
    Process_priv    ENUM('N','Y') NOT NULL DEFAULT 'N',
    Grant_priv    ENUM('N','Y') NOT NULL DEFAULT 'N',
    References_priv  ENUM('N','Y') NOT NULL DEFAULT 'N',
    Alter_priv    ENUM('N','Y') NOT NULL DEFAULT 'N',
    Show_db_priv    ENUM('N','Y') NOT NULL DEFAULT 'N',
    Super_priv    ENUM('N','Y') NOT NULL DEFAULT 'N',
    Create_tmp_table_priv ENUM('N','Y') NOT NULL DEFAULT 'N',
    Lock_tables_priv  ENUM('N','Y') NOT NULL DEFAULT 'N',
    Execute_priv    ENUM('N','Y') NOT NULL DEFAULT 'N',
    Create_view_priv  ENUM('N','Y') NOT NULL DEFAULT 'N',
    Show_view_priv    ENUM('N','Y') NOT NULL DEFAULT 'N',
    Create_routine_priv ENUM('N','Y') NOT NULL DEFAULT 'N',
    Alter_routine_priv ENUM('N','Y') NOT NULL DEFAULT 'N',
    Index_priv    ENUM('N','Y') NOT NULL DEFAULT 'N',
    Create_user_priv  ENUM('N','Y') NOT NULL DEFAULT 'N',
    Event_priv    ENUM('N','Y') NOT NULL DEFAULT 'N',
    Trigger_priv    ENUM('N','Y') NOT NULL DEFAULT 'N',
    PRIMARY KEY (Host, User));
```

```
INSERT INTO mysql.user VALUES ("%", "root", "", "Y", "Y", "Y", "Y", "Y", "Y
    ↪ ", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "
    ↪ Y", "Y", "Y", "Y");
```

6.1.4.1.5 Can TiDB provide services while Loader is running?

TiDB can provide services while Loader is running because Loader inserts the data logically. But do not perform the related DDL operations.

6.1.4.1.6 How to export the data in TiDB?

Currently, TiDB does not support `select into outfile`. You can use the following methods to export the data in TiDB:

- See MySQL uses mysqldump to export part of the table data in Chinese and export data using mysqldump and the WHERE condition.

- Use the MySQL client to export the results of `select` to a file.

6.1.4.1.7 How to migrate from DB2 or Oracle to TiDB?

To migrate all the data or migrate incrementally from DB2 or Oracle to TiDB, see the following solution:

- Use the official migration tool of Oracle, such as OGG, Gateway, CDC (Change Data Capture).
- Develop a program for importing and exporting data.
- Export Spool as text file, and import data using Load infile.
- Use a third-party data migration tool.

Currently, it is recommended to use OGG.

6.1.4.1.8 Error: `java.sql.BatchUpdateExecption:statement count 5001 exceeds the transaction limitation` while using Sqoop to write data into TiDB in batches

In Sqoop, `--batch` means committing 100 `statements` in each batch, but by default each `statement` contains 100 SQL statements. So, 100 * 100 = 10000 SQL statements, which exceeds 5000, the maximum number of statements allowed in a single TiDB transaction.

Two solutions:

- Add the `-Dsqoop.export.records.per.statement=10` option as follows:

```
sqoop export \
    -Dsqoop.export.records.per.statement=10 \
    --connect jdbc:mysql://mysql.example.com/sqoop \
    --username sqoop ${user} \
    --password ${passwd} \
    --table ${tab_name} \
    --export-dir ${dir} \
    --batch
```

- You can also increase the limited number of statements in a single TiDB transaction, but this will consume more memory.

6.1.4.1.9 Does TiDB have a function like the Flashback Query in Oracle? Does it support DDL?

Yes, it does. And it supports DDL as well. For details, see how TiDB reads data from history versions.

6.1.4.2 Migrate the data online

6.1.4.2.1 Syncer

Syncer user guide

See Syncer User Guide.

How to configure to monitor Syncer status?

Download and import Syncer Json to Grafana. Edit the Prometheus configuration file and add the following content:

```
- job_name: 'syncer_ops' // task name
    static_configs:
      - targets: [' 10.10.1.1:10096' ] // Syncer monitoring address and port
          ↪ , informing Prometheus to pull the data of Syncer
```

Restart Prometheus.

Is there a current solution to replicating data from TiDB to other databases like HBase and Elasticsearch?

No. Currently, the data replication depends on the application itself.

Does Syncer support replicating only some of the tables when Syncer is replicating data?

Yes. For details, see Syncer User Guide

Do frequent DDL operations affect the replication speed of Syncer?

Frequent DDL operations may affect the replication speed. For Sycner, DDL operations are executed serially. When DDL operations are executed during data replication, data will be replicated serially and thus the replication speed will be slowed down.

If the machine that Syncer is in is broken and the directory of the `syncer.meta` file is lost, what should I do?

When you replicate data using Syncer GTID, the `syncer.meta` file is constantly updated during the replication process. The current version of Syncer does not contain the design for high availability. The `syncer.meta` configuration file of Syncer is directly stored on the hard disks, which is similar to other tools in the MySQL ecosystem, such as Mydumper.

Two solutions:

- Put the `syncer.meta` file in a relatively secure disk. For example, use disks with RAID 1.
- Restore the location information of history replication according to the monitoring data that Syncer reports to Prometheus regularly. But the location information might be inaccurate due to the delay when a large amount of data is replicated.

If the downstream TiDB data is not consistent with the MySQL data during the replication process of Syncer, will DML operations cause exits?

- If the data exists in the upstream MySQL but does not exist in the downstream TiDB, when the upstream MySQL performs the `UPDATE` or `DELETE` operation on this row of data, Syncer will not report an error and the replication process will not exit, and this row of data does not exist in the downstream.
- If a conflict exists in the primary key indexes or the unique indexes in the downstream, preforming the `UPDATE` operation will cause an exit and performing the `INSERT` operation will not cause an exit.

6.1.4.3 Migrate the traffic

6.1.4.3.1 How to migrate the traffic quickly?

It is recommended to build a multi-source MySQL -> TiDB real-time replication environment using Syncer tool. You can migrate the read and write traffic in batches by editing the network configuration as needed. Deploy a stable network LB (HAproxy, LVS, F5, DNS, etc.) on the upper layer, in order to implement seamless migration by directly editing the network configuration.

6.1.4.3.2 Is there a limit for the total write and read capacity in TiDB?

The total read capacity has no limit. You can increase the read capacity by adding more TiDB servers. Generally the write capacity has no limit as well. You can increase the write capacity by adding more TiKV nodes.

6.1.4.3.3 The error message `transaction too large` is displayed

As distributed transactions need to conduct two-phase commit and the bottom layer performs Raft replication, if a transaction is very large, the commit process would be quite slow and the following Raft replication flow is thus struck. To avoid this problem, we limit the transaction size:

- A transaction is limited to 5000 SQL statements (by default)
- Each Key-Value entry is no more than 6MB
- The total number of Key-Value entry is no more than 300,000 rows
- The total size of Key-Value entry is no more than 100MB

There are similar limits on Google Cloud Spanner.

6.1.4.3.4 How to import data in batches?

When you import data, insert in batches and keep the number of rows within 10,000 for each batch.

6.1.4.3.5 Does TiDB release space immediately after deleting data?

None of the `DELETE`, `TRUNCATE` and `DROP` operations release data immediately. For the `TRUNCATE` and `DROP` operations, after the TiDB GC (Garbage Collection) time (10 minutes by default), the data is deleted and the space is released. For the `DELETE` operation, the data is deleted but the space is not released according to TiDB GC. When subsequent data is written into RocksDB and executes `COMPACT`, the space is reused.

6.1.4.3.6 Can I execute DDL operations on the target table when loading data?

No. None of the DDL operations can be executed on the target table when you load data, otherwise the data fails to be loaded.

6.1.4.3.7 Does TiDB support the `replace into` syntax?

Yes. But the `load data` does not support the `replace into` syntax.

6.1.4.3.8 Why does the query speed getting slow after deleting data?

Deleting a large amount of data leaves a lot of useless keys, affecting the query efficiency. Currently the Region Merge feature is in development, which is expected to solve this problem. For details, see the deleting data section in TiDB Best Practices.

6.1.4.3.9 What is the most efficient way of deleting data?

When deleting a large amount of data, it is recommended to use `Delete * from t where xx limit 5000;`. It deletes through the loop and uses `Affected Rows == 0` as a condition to end the loop, so as not to exceed the limit of transaction size. With the prerequisite of meeting business filtering logic, it is recommended to add a strong filter index column or directly use the primary key to select the range, such as `id >= 5000*n+m and id < 5000*(n+1)+m`.

If the amount of data that needs to be deleted at a time is very large, this loop method will get slower and slower because each deletion traverses backward. After deleting the previous data, lots of deleted flags remain for a short period (then all will be processed by Garbage Collection) and influence the following Delete statement. If possible, it is recommended to refine the Where condition. See details in TiDB Best Practices.

6.1.4.3.10 How to improve the data loading speed in TiDB?

- The Lightning tool is developed for distributed data import. It should be noted that the data import process does not perform a complete transaction process for performance reasons. Therefore, the ACID constraint of the data being imported during the import process cannot be guaranteed. The ACID constraint of the imported data can only be guaranteed after the entire import process ends. Therefore, the applicable scenarios

mainly include importing new data (such as a new table or a new index) or the full backup and restoring (truncate the original table and then import data).
- Data loading in TiDB is related to the status of disks and the whole cluster. When loading data, pay attention to metrics like the disk usage rate of the host, TiClient Error, Backoff, Thread CPU and so on. You can analyze the bottlenecks using these metrics.

6.1.4.3.11 What should I do if it is slow to reclaim storage space after deleting data?

You can configure concurrent GC to increase the speed of reclaiming storage space. The default concurrency is 1, and you can modify it to at most 50% of the number of TiKV instances using the following command:

```
update mysql.tidb set VARIABLE_VALUE="3" where VARIABLE_NAME="
    ↪ tikv_gc_concurrency";
```

6.1.5 SQL optimization

6.1.5.1 TiDB execution plan description

See Understand the Query Execution Plan.

6.1.5.2 Statistics collection

See Introduction to Statistics.

6.1.5.2.1 How to optimize `select count(1)`?

The `count(1)` statement counts the total number of rows in a table. Improving the degree of concurrency can significantly improve the speed. To modify the concurrency, refer to the document. But it also depends on the CPU and I/O resources. TiDB accesses TiKV in every query. When the amount of data is small, all MySQL is in memory, and TiDB needs to conduct a network access.

Recommendations:

1. Improve the hardware configuration. See Software and Hardware Requirements.
2. Improve the concurrency. The default value is 10. You can improve it to 50 and have a try. But usually the improvement is 2-4 times of the default value.
3. Test the `count` in the case of large amount of data.
4. Optimize the TiKV configuration. See Performance Tuning for TiKV.

6.1.5.2.2 How to view the progress of the current DDL job?

You can use `admin show ddl` to view the progress of the current DDL job. The operation is as follows:

```
admin show ddl;
```

```
*************************** 1. row ***************************
  SCHEMA_VER: 140
       OWNER: 1a1c4174-0fcd-4ba0-add9-12d08c4077dc
RUNNING_JOBS: ID:121, Type:add index, State:running, SchemaState:write
    ↪ reorganization, SchemaID:1, TableID:118, RowCount:77312, ArgLen:0,
    ↪ start time: 2018-12-05 16:26:10.652 +0800 CST, Err:<nil>, ErrCount:0,
    ↪  SnapshotVersion:404749908941733890
     SELF_ID: 1a1c4174-0fcd-4ba0-add9-12d08c4077dc
```

From the above results, you can get that the `add index` operation is being processed currently. You can also get from the `RowCount` field of the `RUNNING_JOBS` column that now the `add index` operation has added 77312 rows of indexes.

6.1.5.2.3 How to view the DDL job?

- `admin show ddl`: to view the running DDL job
- `admin show ddl jobs`: to view all the results in the current DDL job queue (including tasks that are running and waiting to run) and the last ten results in the completed DDL job queue
- `admin show ddl job queries 'job_id' [, 'job_id'] ...`: to view the original SQL statement of the DDL task corresponding to the `job_id`; the `job_id` only searches the running DDL job and the last ten results in the DDL history job queue

6.1.5.2.4 Does TiDB support CBO (Cost-Based Optimization)? If yes, to what extent?

Yes. TiDB uses the cost-based optimizer. The cost model and statistics are constantly optimized. TiDB also supports join algorithms like hash join and sort-merge join.

6.1.5.2.5 How to determine whether I need to execute `analyze` on a table?

View the `Healthy` field using `show stats_healthy` and generally you need to execute `analyze` on a table when the field value is smaller than 60.

6.1.5.2.6 What is the ID rule when a query plan is presented as a tree? What is the execution order for this tree?

No rule exists for these IDs but the IDs are unique. When IDs are generated, a counter works and adds one when one plan is generated. The execution order has nothing to do with

the ID. The whole query plan is a tree and the execution process starts from the root node and the data is returned to the upper level continuously. For details about the query plan, see Understanding the TiDB Query Execution Plan.

6.1.5.2.7 In the TiDB query plan, `cop` tasks are in the same root. Are they executed concurrently?

Currently the computing tasks of TiDB belong to two different types of tasks: `cop task` and `root task`.

`cop task` is the computing task which is pushed down to the KV end for distributed execution; `root task` is the computing task for single point execution on the TiDB end.

Generally the input data of `root task` comes from `cop task`; when `root task` processes data, `cop task` of TiKV can processes data at the same time and waits for the pull of `root task` of TiDB. Therefore, `cop` tasks can be considered as executed concurrently; but their data has an upstream and downstream relationship. During the execution process, they are executed concurrently during some time. For example, the first `cop task` is processing the data in [100, 200] and the second `cop task` is processing the data in [1, 100]. For details, see Understanding the TiDB Query Plan.

6.1.6 Database optimization

6.1.6.1 TiDB

6.1.6.1.1 Edit TiDB options

See The TiDB Command Options.

6.1.6.1.2 How to scatter the hotspots?

In TiDB, data is divided into Regions for management. Generally, the TiDB hotspot means the Read/Write hotspot in a Region. In TiDB, for the table whose primary key (PK) is not an integer or which has no PK, you can properly break Regions by configuring `SHARD_ROW_ID_BITS` to scatter the Region hotspots. For details, see the introduction of `SHARD_ROW_ID_BITS` in TiDB Specific System Variables and Syntax.

6.1.6.2 TiKV

6.1.6.2.1 Tune TiKV performance

See Tune TiKV Performance.

6.1.7 Monitor

6.1.7.1 Prometheus monitoring framework

See Overview of the Monitoring Framework.

6.1.7.2 Key metrics of monitoring

See Key Metrics.

6.1.7.2.1 Is there a better way of monitoring the key metrics?

The monitoring system of TiDB consists of Prometheus and Grafana. From the dashboard in Grafana, you can monitor various running metrics of TiDB which include the monitoring metrics of system resources, of client connection and SQL operation, of internal communication and Region scheduling. With these metrics, the database administrator can better understand the system running status, running bottlenecks and so on. In the practice of monitoring these metrics, we list the key metrics of each TiDB component. Generally you only need to pay attention to these common metrics. For details, see Key Metrics.

6.1.7.2.2 The Prometheus monitoring data is deleted every 15 days by default. Could I set it to two months or delete the monitoring data manually?

Yes. Find the startup script on the machine where Prometheus is started, edit the startup parameter and restart Prometheus.

```
--storage.tsdb.retention="60d"
```

6.1.7.2.3 Region Health monitor

In TiDB 2.0, Region health is monitored in the PD metric monitoring page, in which the `Region Health` monitoring item shows the statistics of all the Region replica status. `miss` means shortage of replicas and `extra` means the extra replica exists. In addition, `Region Health` also shows the isolation level by `label`. `level-1` means the Region replicas are isolated physically in the first `label` level. All the Regions are in `level-0` when `location label` is not configured.

6.1.7.2.4 What is the meaning of `selectsimplefull` in Statement Count monitor?

It means full table scan but the table might be a small system table.

6.1.7.2.5 What is the difference between `QPS` and `Statement OPS` in the monitor?

The `QPS` statistics is about all the SQL statements, including `use database`, `load data`, `begin`, `commit`, `set`, `show`, `insert` and `select`.

The `Statement OPS` statistics is only about applications related SQL statements, including `select`, `update` and `insert`, therefore the `Statement OPS` statistics matches the applications better.

6.1.8 Deployment on the cloud

6.1.8.1 Public cloud

6.1.8.1.1 What cloud vendors are currently supported by TiDB?

TiDB supports deployment on Google GKE, AWS EKS and Alibaba Cloud ACK.

In addition, TiDB is currently available on JD Cloud and UCloud, and has the first-level database entries on them.

6.1.9 Troubleshoot

6.1.9.1 TiDB custom error messages

6.1.9.1.1 ERROR 8005 (HY000): Write Conflict, txnStartTS is stale

Check whether `tidb_disable_txn_auto_retry` is set to `on`. If so, set it to `off`; if it is already `off`, increase the value of `tidb_retry_limit` until the error no longer occurs.

6.1.9.1.2 ERROR 9001 (HY000): PD Server Timeout

A PD request timeout. Check the status, monitoring data and log of the PD server, and the network between the TiDB server and the PD server.

6.1.9.1.3 ERROR 9002 (HY000): TiKV Server Timeout

A TiKV request timeout. Check the status, monitoring data and log of the TiKV server, and the network between the TiDB server and the TiKV server.

6.1.9.1.4 ERROR 9003 (HY000): TiKV Server is Busy

The TiKV server is busy. This usually occurs when the database load is very high. Check the status, monitoring data and log of the TiKV server.

6.1.9.1.5 ERROR 9004 (HY000): Resolve Lock Timeout

A lock resolving timeout. This usually occurs when a large number of transaction conflicts exist. Check the application code to see whether lock contention exists in the database.

6.1.9.1.6 ERROR 9005 (HY000): Region is unavailable

The accessed Region is not available. A Raft Group is not available, with possible reasons like an inadequate number of replicas. This usually occurs when the TiKV server is busy or the TiKV node is shut down. Check the status, monitoring data and log of the TiKV server.

6.1.9.1.7 ERROR 9006 (HY000): GC life time is shorter than transaction duration

The interval of `GC Life Time` is too short. The data that should have been read by long transactions might be deleted. You can add `GC Life Time` using the following command:

```
update mysql.tidb set variable_value='30m' where variable_name='
    ↪ tikv_gc_life_time';
```

> **Note:**
>
> "30m" means only cleaning up the data generated 30 minutes ago, which might consume some extra storage space.

6.1.9.1.8 ERROR 9007 (HY000): Write Conflict

Check whether `tidb_disable_txn_auto_retry` is set to `on`. If so, set it to `off`; if it is already `off`, increase the value of `tidb_retry_limit` until the error no longer occurs.

6.1.9.2 MySQL native error messages

6.1.9.2.1 ERROR 2013 (HY000): Lost connection to MySQL server during query

- Check whether panic is in the log.
- Check whether OOM exists in dmesg using `dmesg -T | grep -i oom`.
- A long time of no access might also lead to this error. It is usually caused by TCP timeout. If TCP is not used for a long time, the operating system kills it.

6.1.9.2.2 ERROR 1105 (HY000): other error: unknown error Wire Error(InvalidEnumValue(4004))

This error usually occurs when the version of TiDB does not match with the version of TiKV. To avoid version mismatch, upgrade all components when you upgrade the version.

6.1.9.2.3 ERROR 1148 (42000): the used command is not allowed with this TiDB version

When you execute the `LOAD DATA LOCAL` statement but the MySQL client does not allow executing this statement (the value of the `local_infile` option is 0), this error occurs.

The solution is to use the `--local-infile=1` option when you start the MySQL client. For example, use command like `mysql --local-infile=1 -u root -h 127.0.0.1 -P ↪ 4000`. The default value of `local-infile` is different in different versions of MySQL

client, therefore you need to configure it in some MySQL clients and do not need to configure it in some others.

6.1.9.2.4 ERROR 9001 (HY000): PD server timeout start timestamp may fall behind safe point

This error occurs when TiDB fails to access PD. A worker in the TiDB background continuously queries the safepoint from PD and this error occurs if it fails to query within 100s. Generally, it is because the disk on PD is slow and busy or the network failed between TiDB and PD. For the details of common errors, see Error Number and Fault Diagnosis.

6.1.9.3 TiDB log error messages

6.1.9.3.1 EOF error

When the client or proxy disconnects from TiDB, TiDB does not immediately notice that the connection has been disconnected. Instead, TiDB can only notice the disconnection when it begins to return data to the connection. At this time, the log prints an EOF error.

6.2 TiDB Lightning FAQ

6.2.1 What is the minimum TiDB/TiKV/PD cluster version supported by Lightning?

The version of TiDB Lightning should be the same as the cluster. The earliest available version is 2.0.9, but we recommend using the latest stable version (3.0).

6.2.2 Does TiDB Lightning support importing multiple schemas (databases)?

Yes.

6.2.3 What is the privilege requirements for the target database?

TiDB Lightning requires the following privileges:

- SELECT
- UPDATE
- ALTER
- CREATE
- DROP

If the TiDB-backend is chosen, or the target database is used to store checkpoints, it additionally requires these privileges:

- INSERT
- DELETE

The Importer-backend does not require these two privileges because data is ingested into TiKV directly, which bypasses the entire TiDB privilege system. This is secure as long as the ports of TiKV, TiKV Importer and TiDB Lightning are not reachable outside the cluster.

If the `checksum` configuration of TiDB Lightning is set to `true`, then the admin user privileges in the downstream TiDB need to be granted to TiDB Lightning.

6.2.4 TiDB Lightning encountered an error when importing one table. Will it affect other tables? Will the process be terminated?

If only one table has an error encountered, the rest will still be processed normally.

6.2.5 How to properly restart TiDB Lightning?

Depending on the status of `tikv-importer`, the basic sequence of restarting TiDB Lightning is like this:

If `tikv-importer` is still running:

1. Stop `tidb-lightning`.
2. Perform the intended modifications, such as fixing the source data, changing settings, replacing hardware etc.
3. If the modification previously has changed any table, remove the corresponding checkpoint too.
4. Start `tidb-lightning`.

If `tikv-importer` needs to be restarted:

1. Stop `tidb-lightning`.
2. Stop `tikv-importer`.
3. Perform the intended modifications, such as fixing the source data, changing settings, replacing hardware etc.
4. Start `tikv-importer`.
5. Start `tidb-lightning` *and wait until the program fails with CHECKSUM error, if any.*
    - Restarting `tikv-importer` would destroy all engine files still being written, but `tidb-lightning` did not know about it. As of v3.0 the simplest way is to let `tidb-lightning` go on and retry.
6. Destroy the failed tables and checkpoints
7. Start `tidb-lightning` again.

6.2.6 How to ensure the integrity of the imported data?

TiDB Lightning by default performs checksum on the local data source and the imported tables. If there is checksum mismatch, the process would be aborted. These checksum information can be read from the log.

You could also execute the `ADMIN CHECKSUM TABLE` SQL command on the target table to recompute the checksum of the imported data.

```
ADMIN CHECKSUM TABLE `schema`.`table`;
```

```
+---------+------------+--------------------+-----------+-------------+
| Db_name | Table_name | Checksum_crc64_xor | Total_kvs | Total_bytes |
+---------+------------+--------------------+-----------+-------------+
| schema | table     | 5505282386844578743 |     3 |         96 |
+---------+------------+--------------------+-----------+-------------+
1 row in set (0.01 sec)
```

6.2.7 What kind of data source format is supported by Lightning?

TiDB Lightning only supports the SQL dump generated by Mydumper or CSV files stored in the local file system.

6.2.8 Could TiDB Lightning skip creating schema and tables?

Yes. If you have already created the tables in the target database, you could set `no-schema = true` in the `[mydumper]` section in `tidb-lightning.toml`. This makes TiDB Lightning skip the `CREATE TABLE` invocations and fetch the metadata directly from the target database. TiDB Lightning will exit with error if a table is actually missing.

6.2.9 Can the Strict SQL Mode be disabled to allow importing invalid data?

Yes. By default, the `sql_mode` used by TiDB Lightning is `"STRICT_TRANS_TABLES, NO_ENGINE_SUBSTITUTION"`, which disallows invalid data such as the date `1970-00-00`. The mode can be changed by modifying the `sql-mode` setting in the `[tidb]` section in `tidb-lightning.toml`.

```
...
[tidb]
sql-mode = ""
...
```

6.2.10 Can one `tikv-importer` serve multiple `tidb-lightning` instances?

Yes, as long as every `tidb-lightning` instance operates on different tables.

6.2.11 How to stop the `tikv-importer` process?

To stop the `tikv-importer` process, you can choose the corresponding operation according to your deployment method.

- For deployment using TiDB Ansible: run `scripts/stop_importer.sh` on the Importer server.
- For manual deployment: if `tikv-importer` is running in foreground, press Ctrl+C to exit. Otherwise, obtain the process ID using the `ps aux | grep tikv-importer` command and then terminate the process using the `kill «pid»` command.

6.2.12 How to stop the `tidb-lightning` process?

To stop the `tidb-lightning` process, you can choose the corresponding operation according to your deployment method.

- For deployment using TiDB Ansible: run `scripts/stop_lightning.sh` on the Lightning server.
- For manual deployment: if `tidb-lightning` is running in foreground, press Ctrl+C to exit. Otherwise, obtain the process ID using the `ps aux | grep tidb-lighting` command and then terminate the process using the `kill -2 «pid»` command.

6.2.13 Why the `tidb-lightning` process suddenly quits while running in background?

It is potentially caused by starting `tidb-lightning` incorrectly, which causes the system to send a SIGHUP signal to stop the `tidb-lightning` process. In this situation, `tidb-lightning.log` usually outputs the following log:

```
[2018/08/10 07:29:08.310 +08:00] [INFO] [main.go:41] ["got signal to exit"]
    ↪ [signal=hangup]
```

It is not recommended to directly use `nohup` in the command line to start `tidb-lightning`. You can start `tidb-lightning` by executing a script.

6.2.14 Why my TiDB cluster is using lots of CPU resources and running very slowly after using TiDB Lightning?

If `tidb-lightning` abnormally exited, the cluster might be stuck in the "import mode", which is not suitable for production. You can force the cluster back to "normal mode" using the following command:

```
tidb-lightning-ctl --switch-mode=normal
```

6.2.15 Can TiDB Lightning be used with 1-Gigabit network card?

The TiDB Lightning toolset is best used with a 10-Gigabit network card. 1-Gigabit network cards are *not recommended*, especially for `tikv-importer`.

1-Gigabit network cards can only provide a total bandwidth of 120 MB/s, which has to be shared among all target TiKV stores. TiDB Lightning can easily saturate all bandwidth of the 1-Gigabit network and bring down the cluster because PD is unable to be contacted anymore. To avoid this, set an *upload speed limit* in Importer's configuration:

```
[import]
## Restricts the total upload speed to TiKV to 100 MB/s or less
upload-speed-limit = "100MB"
```

6.2.16 Why TiDB Lightning requires so much free space in the target TiKV cluster?

With the default settings of 3 replicas, the space requirement of the target TiKV cluster is 6 times the size of data source. The extra multiple of "2" is a conservative estimation because the following factors are not reflected in the data source:

- The space occupied by indices
- Space amplification in RocksDB

6.2.17 Can TiKV Importer be restarted while TiDB Lightning is running?

No. Importer stores some information of engines in memory. If `tikv-importer` is restarted, `tidb-lightning` will be stopped due to lost connection. At this point, you need to destroy the failed checkpoints as those Importer-specific information is lost. You can restart Lightning afterwards.

See also How to properly restart TiDB Lightning? for the correct sequence.

6.2.18 How to completely destroy all intermediate data associated with TiDB Lightning?

1. Delete the checkpoint file.

   ```
   tidb-lightning-ctl --config conf/tidb-lightning.toml --checkpoint-
   ↪ remove=all
   ```

   If, for some reason, you cannot run this command, try manually deleting the file `/tmp
   ↪ /tidb_lightning_checkpoint.pb`.

2. If you are using Local-backend, delete the `sorted-kv-dir` directory in the configuration. If you are using Importer-backend, delete the entire `import` directory on the machine hosting `tikv-importer`.

3. Delete all tables and databases created on the TiDB cluster, if needed.

6.2.19 Why does TiDB Lightning report the `could not find first pair, this shouldn't happen` error?

This error occurs possibly because the number of files opened by TiDB Lightning exceeds the system limit when TiDB Lightning reads the sorted local files. In the Linux system, you can use the `ulimit -n` command to confirm whether the value of this system limit is too small. It is recommended that you adjust this value to `1000000` (`ulimit -n 1000000`) during TiDB Lightning import.

6.2.20 Import speed is too slow

Normally it takes TiDB Lightning 2 minutes per thread to import a 256 MB data file. If the speed is much slower than this, there is an error. You can check the time taken for each data file from the log mentioning `restore chunk ⋯ takes`. This can also be observed from metrics on Grafana.

There are several reasons why TiDB Lightning becomes slow:

Cause 1: `region-concurrency` is set too high, which causes thread contention and reduces performance.

1. The setting can be found from the start of the log by searching `region-concurrency`.
2. If TiDB Lightning shares the same machine with other services (for example, TiKV Importer), `region-concurrency` must be **manually** set to 75% of the total number of CPU cores.
3. If there is a quota on CPU (for example, limited by Kubernetes settings), TiDB Lightning may not be able to read this out. In this case, `region-concurrency` must also be **manually** reduced.

Cause 2: The table schema is too complex.

Every additional index introduces a new KV pair for each row. If there are N indices, the actual size to be imported would be approximately (N+1) times the size of the Dumpling output. If the indices are negligible, you may first remove them from the schema, and add them back using `CREATE INDEX` after the import is complete.

Cause 3: Each file is too large.

TiDB Lightning works the best when the data source is broken down into multiple files of size around 256 MB so that the data can be processed in parallel. If each file is too large, TiDB Lightning might not respond.

If the data source is CSV, and all CSV files have no fields containing newline control characters (U+000A and U+000D), you can turn on "strict format" to let TiDB Lightning automatically split the large files.

```
[mydumper]
strict-format = true
```

Cause 4: TiDB Lightning is too old.

Try the latest version! Maybe there is new speed improvement.

6.2.21 `checksum failed: checksum mismatched remote vs local`

Cause: The checksum of a table in the local data source and the remote imported database differ. This error has several deeper reasons:

1. The table might already have data before. These old data can affect the final checksum.
2. If the remote checksum is 0, which means nothing is imported, it is possible that the cluster is too hot and fails to take in any data.
3. If the data is mechanically generated, ensure it respects the constrains of the table:
    - `AUTO_INCREMENT` columns need to be positive, and do not contain the value "0".
    - The UNIQUE and PRIMARY KEYs must have no duplicated entries.
4. If TiDB Lightning has failed before and was not properly restarted, a checksum mismatch may happen due to data being out-of-sync.

Solutions:

1. Delete the corrupted data using `tidb-lightning-ctl`, and restart TiDB Lightning to import the affected tables again.

    ```
    tidb-lightning-ctl --config conf/tidb-lightning.toml --checkpoint-error
        ↪ -destroy=all
    ```

2. Consider using an external database to store the checkpoints (change `[checkpoint]` ↪ `dsn`) to reduce the target database's load.
3. If TiDB Lightning was improperly restarted, see also the "How to properly restart TiDB Lightning" section in the FAQ.

6.2.22 `Checkpoint for … has invalid status:` (error code)

Cause: Checkpoint is enabled, and TiDB Lightning or TiKV Importer has previously abnormally exited. To prevent accidental data corruption, Lightning will not start until the error is addressed.

The error code is an integer smaller than 25, with possible values of 0, 3, 6, 9, 12, 14, 15, 17, 18, 20, and 21. The integer indicates the step where the unexpected exit occurs in the import process. The larger the integer is, the later step the exit occurs at.

Solutions:

If the error was caused by invalid data source, delete the imported data using `tidb-lightning-ctl` and start Lightning again.

```
tidb-lightning-ctl --config conf/tidb-lightning.toml --checkpoint-error-destroy=all
```

See the Checkpoints control section for other options.

6.2.23 `ResourceTemporarilyUnavailable("Too many open engines …: …")`

Cause: The number of concurrent engine files exceeds the limit specified by `tikv-importer`. This could be caused by misconfiguration. Additionally, if `tidb-lightning` exited abnormally, an engine file might be left at a dangling open state, which could cause this error as well.

Solutions:

1. Increase the value of `max-open-engines` setting in `tikv-importer.toml`. This value is typically dictated by the available memory. This could be calculated by using:

   Max Memory Usage ≈ `max-open-engines` × `write-buffer-size` × `max-write-buffer-number`

2. Decrease the value of `table-concurrency` + `index-concurrency` so it is less than `max-open-engines`.

3. Restart `tikv-importer` to forcefully remove all engine files (default to `./data.import/`). This also removes all partially imported tables, which requires Lightning to clear the outdated checkpoints.

   ```
   tidb-lightning-ctl --config conf/tidb-lightning.toml --checkpoint-error-destroy=all
   ```

6.2.24 `cannot guess encoding for input file, please convert to UTF-8 manually`

Cause: TiDB Lightning only recognizes the UTF-8 and GB-18030 encodings for the table schemas. This error is emitted if the file isn't in any of these encodings. It is also possible that the file has mixed encoding, such as containing a string in UTF-8 and another string in GB-18030, due to historical `ALTER TABLE` executions.

Solutions:

1. Fix the schema so that the file is entirely in either UTF-8 or GB-18030.
2. Manually `CREATE` the affected tables in the target database, and then set `[mydumper] no-schema = true` to skip automatic table creation.
3. Set `[mydumper] character-set = "binary"` to skip the check. Note that this might introduce mojibake into the target database.

6.2.25 `[sql2kv] sql encode error = [types:1292]invalid time format: '{1970 1 1 …}'`

Cause: A table contains a column with the `timestamp` type, but the time value itself does not exist. This is either because of DST changes or the time value has exceeded the supported range (Jan 1, 1970 to Jan 19, 2038).

Solutions:

1. Ensure Lightning and the source database are using the same time zone.

   When executing Lightning directly, the time zone can be forced using the `$TZ` environment variable.

   ```
   # Manual deployment, and force Asia/Shanghai.
   TZ='Asia/Shanghai' bin/tidb-lightning -config tidb-lightning.toml
   ```

2. When exporting data using Mydumper, make sure to include the `--skip-tz-utc` flag.
3. Ensure the entire cluster is using the same and latest version of `tzdata` (version 2018i or above).

   On CentOS, run `yum info tzdata` to check the installed version and whether there is an update. Run `yum upgrade tzdata` to upgrade the package.

6.2.26 `[Error 8025: entry too large, the max entry size is 6291456]`

Cause: A single row of key-value pairs generated by TiDB Lightning exceeds the limit set by TiDB.

Solution:

Currently, the limitation of TiDB cannot be bypassed. You can only ignore this table to ensure the successful import of other tables.

6.2.27 `restore table test.district failed: unknown columns in header [...]`

This error occurs usually because the CSV data file does not contain a header (the first row is not column names but data). Therefore, you need to add the following configuration to the TiDB Lightning configuration file:

```
[mydumper.csv]
header = false
```

6.3 FAQs After Upgrade

This document lists some FAQs and their solutions after you upgrade TiDB.

6.3.1 The character set (charset) errors when executing DDL operations

In v2.1.0 and earlier versions (including all versions of v2.0), the character set of TiDB is UTF-8 by default. But starting from v2.1.1, the default character set has been changed into UTF8MB4.

If you explicitly specify the charset of a newly created table as UTF-8 in v2.1.0 or earlier versions, then you might fail to execute DDL operations after upgrading TiDB to v2.1.1.

To avoid this issue, you need to pay attention to:

- Point #1: before v2.1.3, TiDB does not support modifying the charset of the column. Therefore, when you execute DDL operations, you need to make sure that the charset of the new column is consistent with that of the original column.
- Point #2: before v2.1.3, even if the charset of the column is different from that of the table, `show create table` does not show the charset of the column. But as shown in the following example, you can view it by obtaining the metadata of the table through the HTTP API.

6.3.1.1 Issue #1: `unsupported modify column charset utf8mb4 not match origin utf8`

- Before upgrading, the following operations are executed in v2.1.0 and earlier versions.

```
create table t(a varchar(10)) charset=utf8;
```

```
Query OK, 0 rows affected
Time: 0.106s
```

```
show create table t;
```

```
+-------+-------------------------------------------------------+
| Table | Create Table                                          |
+-------+-------------------------------------------------------+
| t     | CREATE TABLE `t` (                                    |
|       |   `a` varchar(10) DEFAULT NULL                        |
|       | ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin |
+-------+-------------------------------------------------------+
1 row in set
Time: 0.006s
```

- After upgrading, the following error is reported in v2.1.1 and v2.1.2 but there is no such error in v2.1.3 and the later versions.

```
alter table t change column a a varchar(20);
```

```
ERROR 1105 (HY000): unsupported modify column charset utf8mb4 not match
    ↪ origin utf8
```

Solution:

You can explicitly specify the column charset as the same with the original charset.

```
alter table t change column a a varchar(22) character set utf8;
```

- According to Point #1, if you do not specify the column charset, UTF8MB4 is used by default, so you need to specify the column charset to make it consistent with the original one.
- According to Point #2, you can obtain the metadata of the table through the HTTP API, and find the column charset by searching the column name and the keyword "Charset".

```
curl "http://$IP:10080/schema/test/t" | python -m json.tool
```

Here the python tool is used to format JSON, which is not required and only for the convenience to add comments.

```
{
    "ShardRowIDBits": 0,
    "auto_inc_id": 0,
    "charset": "utf8", # The charset of the table.
    "collate": "",
    "cols": [          # The relevant information about the columns.
        {
            ...
```

```
            "id": 1,
            "name": {
                "L": "a",
                "O": "a"  # The column name.
            },
            "offset": 0,
            "origin_default": null,
            "state": 5,
            "type": {
                "Charset": "utf8", # The charset of column a.
                "Collate": "utf8_bin",
                "Decimal": 0,
                "Elems": null,
                "Flag": 0,
                "Flen": 10,
                "Tp": 15
            }
        }
    ],
    ...
}
```

6.3.1.2 Issue #2: unsupported modify charset from utf8mb4 to utf8

- Before upgrading, the following operations are executed in v2.1.1 and v2.1.2.

```
create table t(a varchar(10)) charset=utf8;
```

```
Query OK, 0 rows affected
Time: 0.109s
```

```
show create table t;
```

```
+-------+-------------------------------------------------------+
| Table | Create Table                                          |
+-------+-------------------------------------------------------+
| t     | CREATE TABLE `t` (                                    |
|       |   `a` varchar(10) DEFAULT NULL                        |
|       | ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin |
+-------+-------------------------------------------------------+
```

In the above example, `show create table` only shows the charset of the table, but the charset of the column is actually UTF8MB4, which can be confirmed by obtaining the schema through the HTTP API. However, when a new table is created, the charset

of the column should stay consistent with that of the table. This bug has been fixed in v2.1.3.

- After upgrading, the following operations are executed in v2.1.3 and the later versions.

```
show create table t;
```

```
+-------+-------------------------------------------------------------------------+
    ↪
| Table | Create Table                                                            |
+-------+-------------------------------------------------------------------------+
    ↪
| t     | CREATE TABLE `t` (                                                      |
|       |   `a` varchar(10) CHARSET utf8mb4 COLLATE utf8mb4_bin DEFAULT
    ↪ NULL |
|       | ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin                   |
+-------+-------------------------------------------------------------------------+
    ↪
1 row in set
Time: 0.007s
```

```
alter table t change column a a varchar(20);
```

```
ERROR 1105 (HY000): unsupported modify charset from utf8mb4 to utf8
```

Solution:

- Starting from v2.1.3, TiDB supports modifying the charsets of the column and the table, so it is recommended to modify the table charset into UTF8MB4.

```
alter table t convert to character set utf8mb4;
```

- You can also specify the column charset as done in Issue #1, making it stay consistent with the original column charset (UTF8MB4).

```
alter table t change column a a varchar(20) character set utf8mb4;
```

6.3.1.3 Issue #3: ERROR 1366 (HY000): incorrect utf8 value f09f8c80( ) for column a

In TiDB v2.1.1 and earlier versions, if the charset is UTF-8, there is no UTF-8 Unicode encoding check on the inserted 4-byte data. But in v2.1.2 and the later versions, this check is added.

- Before upgrading, the following operations are executed in v2.1.1 and earlier versions.

```
create table t(a varchar(100) charset utf8);
```

```
Query OK, 0 rows affected
```

```
insert t values (unhex('f09f8c80'));
```

```
Query OK, 1 row affected
```

- After upgrading, the following error is reported in v2.1.2 and the later versions.

```
insert t values (unhex('f09f8c80'));
```

```
ERROR 1366 (HY000): incorrect utf8 value f09f8c80( ) for column a
```

Solution:

- In v2.1.2: this version does not support modifying the column charset, so you have to skip the UTF-8 check.

```
set @@session.tidb_skip_utf8_check=1;
```

```
Query OK, 0 rows affected
```

```
insert t values (unhex('f09f8c80'));
```

```
Query OK, 1 row affected
```

- In v2.1.3 and the later versions: it is recommended to modify the column charset into UTF8MB4. Or you can set `tidb_skip_utf8_check` to skip the UTF-8 check. But if you skip the check, you might fail to replicate data from TiDB to MySQL because MySQL executes the check.

```
alter table t change column a a varchar(100) character set utf8mb4;
```

```
Query OK, 0 rows affected
```

```
insert t values (unhex('f09f8c80'));
```

```
Query OK, 1 row affected
```

Specifically, you can use the variable `tidb_skip_utf8_check` to skip the legal UTF-8 and UTF8MB4 check on the data. But if you skip the check, you might fail to replicate the data from TiDB to MySQL because MySQL executes the check.

If you only want to skip the UTF-8 check, you can set `tidb_check_mb4_value_in_utf8` ↪ . This variable is added to the `config.toml` file in v2.1.3, and you can modify `check-mb4-value-in-utf8` in the configuration file and then restart the cluster to enable it.

Starting from v2.1.5, you can set `tidb_check_mb4_value_in_utf8` through the HTTP API and the session variable:

- HTTP API (the HTTP API can be enabled only on a single server)
  - To enable HTTP API:

```
curl -X POST -d "check_mb4_value_in_utf8=1" http://{TiDBIP
    ↪ }:10080/settings
```

  - To disable HTTP API:

```
curl -X POST -d "check_mb4_value_in_utf8=0" http://{TiDBIP
    ↪ }:10080/settings
```

- Session variable
  - To enable session variable:

```
set @@session.tidb_check_mb4_value_in_utf8 = 1;
```

  - To disable session variable:

```
set @@session.tidb_check_mb4_value_in_utf8 = 0;
```

7 Support

7.1 Support Resources

You can reach out to the community members via any one of the following ways:

- Slack Channel: https://pingcap.com/tidbslack
- Google Groups: https://groups.google.com/forum/#!forum/tidb-user
- Stack Overflow: https://stackoverflow.com/questions/tagged/tidb
- Twitter: https://twitter.com/PingCAP
- Facebook: https://www.facebook.com/pingcap2015
- Reddit: https://www.reddit.com/r/TiDB
- GitHub: https://github.com/pingcap/tidb/issues

Please contact us for more information on Enterprise supports and cooperations.

7.2 Report an Issue

We strive to make TiDB compatible with MySQL 5.7. If you discover a difference in behavior that is not documented, a bug, or strange performance characteristics please create a new issue in GitHub.

You may use GitHub issues for Bug Reports, Feature Requests, General Questions and Performance Questions. Please make sure that you fill out the issue template completely so we can respond appropriately.

8 Contribute

8.1 Contribute

8.1.1 Contribute to TiDB

We recommend starting with an existing issue with the help-wanted label. These issues are well suited for new contributors.

If a PR (Pull Request) submitted to the TiDB/TiKV/TiSpark/PD/TiDB Operator/Docs/Docs-cn projects by you is approved and merged, then you become a TiDB Contributor.

Before submitting a pull request, sign the CLA.

If you want to work on a new idea of relatively small scope:

1. Submit an issue describing your proposed change to the repo in question.
2. The repo owners will respond to your issue promptly.
3. If your proposed change is accepted, sign the CLA, and start work in your fork.
4. Submit a pull request containing a tested change.

Contributions are welcomed and greatly appreciated. See CONTRIBUTING.md for details on submitting patches and the contribution workflow.

8.1.2 Improve the docs

We welcome contributions to help improve the documentation!

The TiDB docs are written in Markdown and use the Google Developer Documentation Style Guide. Don't worry if this is new to you, we are happy to guide you along the way.

To contribute, please fork the docs repository and send a Pull Request.

9 Releases

9.1 TiDB Release Notes

9.1.1 3.0

- 3.0.20
- 3.0.19
- 3.0.18
- 3.0.17
- 3.0.16
- 3.0.15
- 3.0.14
- 3.0.13
- 3.0.12
- 3.0.11
- 3.0.10
- 3.0.9
- 3.0.8
- 3.0.7
- 3.0.6
- 3.0.5
- 3.0.4
- 3.0.3
- 3.0.2
- 3.0.1
- 3.0 GA
- 3.0.0-rc.3
- 3.0.0-rc.2
- 3.0.0-rc.1
- 3.0.0-beta.1
- 3.0.0-beta

9.1.2 2.1

- 2.1.19
- 2.1.18
- 2.1.17
- 2.1.16
- 2.1.15
- 2.1.14
- 2.1.13
- 2.1.12
- 2.1.11

- 2.1.10
- 2.1.9
- 2.1.8
- 2.1.7
- 2.1.6
- 2.1.5
- 2.1.4
- 2.1.3
- 2.1.2
- 2.1.1
- 2.1 GA
- 2.1 RC5
- 2.1 RC4
- 2.1 RC3
- 2.1 RC2
- 2.1 RC1
- 2.1 Beta

9.1.3 2.0

- 2.0.11
- 2.0.10
- 2.0.9
- 2.0.8
- 2.0.7
- 2.0.6
- 2.0.5
- 2.0.4
- 2.0.3
- 2.0.2
- 2.0.1
- 2.0
- 2.0 RC5
- 2.0 RC4
- 2.0 RC3
- 2.0 RC1
- 1.1 Beta
- 1.1 Alpha

9.1.4 1.0

- 1.0.8
- 1.0.7
- 1.0.6

9.2 v3.0

9.2.1 TiDB 3.0.20 Release Notes

Release date: December 25, 2020

TiDB version: 3.0.20

9.2.1.1 Compatibility Change

- TiDB
  - Deprecate the `enable-streaming` configuration item #21054

9.2.1.2 Improvements

- TiDB
  - Raise an error when preparing the `LOAD DATA` statement #21222
- TiKV
  - Add the `end_point_slow_log_threshold` configuration item #9145

9.2.1.3 Bug Fixes

- TiDB
  - Fix the incorrect cache of the transaction status for pessimistic transactions #21706
  - Fix the issue of inaccurate statistics that occurs when querying `INFORMATION_SCHEMA` ↪ `.TIDB_HOT_REGIONS` #21319

- Fix the issue that `DELETE` might not delete data correctly when the database name is not in a pure lower representation #21205
- Fix the issue of stack overflow that occurs when building the recursive view #21000
- Fix the issue of goroutine leak in TiKV client #20863
- Fix the wrong default zero value for the `year` type #20828
- Fix the issue of goroutine leak in index lookup join #20791
- Fix the issue that executing `INSERT SELECT FOR UPDATE` returns the malformed packet in the pessimistic transaction #20681
- Fix the unknown time zone `'posixrules'` #20605
- Fix the issue that occurs when converting the unsigned integer type to the bit type #20362
- Fix the corrupted default value of the bit type column #20339
- Fix the potentially incorrect results when one of the equal condition is the `Enum` or `Set` type #20296
- Fix a wrong behavior of `!= any()` #20061
- Fix the issue that type conversion in `BETWEEN...AND...` returns invalid results #21503
- Fix a compatibility issue with the `ADDDATE` function #21008
- Set the correct default value for newly added `Enum` column #20999
- Fix the result of SQL statements like `SELECT DATE_ADD('2007-03-28 22:08:28',INTERVAL "-2.-2" SECOND)` to be compatible with MySQL #20627
- Fix the incorrect default value when modifying the column type #20532
- Fix the issue that the `timestamp` function gets wrong result when the input argument is the `float` or `decimal` type #20469
- Fix a potential deadlock issue in statistics #20424
- Fix the issue that the overflown float type data is inserted #20251

- TiKV
  - Fix the issue that an error is returned indicating that a key exists when this key is locked and deleted in a committed transaction #8931

- PD
  - Fix the issue that too many logs are printed when starting PD and when there are too many stale Regions #3064

9.2.2 TiDB 3.0.19 Release Notes

Release date: September 25, 2020

TiDB version: 3.0.19

9.2.2.1 Compatibility Changes

- PD
  - Change the import path from `pingcap/pd` to `tikv/pd` #2779
  - Change the copyright information from `PingCAP, Inc` to `TiKV Project Authors` #2777

9.2.2.2 Improvements

- TiDB
  - Mitigate the impact of failure recovery on QPS performance #19764
  - Support adjusting the concurrency of the `union` operator #19885
- TiKV
  - Set `sync-log` to `true` as a nonadjustable value #8636
- PD
  - Add an alert rule for PD restart #2789

9.2.2.3 Bug Fixes

- TiDB
  - Fix the query error that occurs when the `slow-log` file does not exist #20050
  - Add the privilege check for `SHOW STATS_META` and `SHOW STATS_BUCKET` #19759
  - Forbid changing the decimal type to the integer type #19681
  - Fix the issue that the constraint is not checked when altering the `ENUM`/`SET` type column #20045
  - Fix the bug that tidb-server does not release table locks after a panic #20021
  - Fix the bug that the `OR` operator is not handled correctly in the `WHERE` clause #19901
- TiKV
  - Fix the bug that TiKV panics when parsing responses with missing reason phrases #8540
- Tools
  - TiDB Lightning
    * Fix the issue that the TiDB Lightning process does not exit in time when encountering illegal UTF characters in CSV in the strict mode #378

9.2.3 TiDB 3.0.18 Release Notes

Release date: August 21, 2020

TiDB version: 3.0.18

9.2.3.1 Improvements

- Tools
  - TiDB Binlog
    * Support the time duration format of Go for the Pump GC configuration #996

9.2.3.2 Bug Fixes

- TiDB
  - Fix the issue that the wrong handling of the `decimal` type by the `Hash` function causes the wrong HashJoin result #19185
  - Fix the issue that the wrong handling of the `set` and `enum` types by the `Hash` function causes the wrong HashJoin result #19175
  - Fix the issue that the check for duplicate keys fails in the pessimistic locking mode #19236
  - Fix the issue that the `Apply` and `Union Scan` operators cause the wrong execution result #19297
  - Fix the issue that some cached execution plans are incorrectly executed in transaction #19274
- TiKV
  - Change the GC failure log from `error` to the `warning` level #8444
- Tools
  - TiDB Lightning
    * Fix the issue that the `--log-file` argument does not take effect #345
    * Fix the syntax error on empty binary/hex literals when using TiDB-backend #357
    * Fix the unexpected `switch-mode` call when using TiDB-backend #368

9.2.4 TiDB 3.0.17 Release Notes

Release date: Aug 3, 2020

TiDB version: 3.0.17

9.2.4.1 Improvements

- TiDB
  - Decrease the default value of the `query-feedback-limit` configuration item from 1024 to 512, and improve the statistics feedback mechanism to ease its impact on the cluster #18770
  - Limit batch split count for one request #18694
  - Accelerate `/tiflash/replica` HTTP API when there are many history DDL jobs in the TiDB cluster #18386
  - Improve row count estimation for index equal condition #17609
  - Speed up the execution of `kill tidb conn_id` #18506
- TiKV
  - Add the `hibernate-timeout` configuration that delays region hibernation to improve rolling update performance #8207
- Tools
  - TiDB Lightning
    * `[black-white-list]` has been deprecated with a newer, easier-to-understand filter format #332

9.2.4.2 Bug Fixes

- TiDB
  - Return the actual error message instead of an empty set when a query which contains `IndexHashJoin` or `IndexMergeJoin` encounters a panic #18498
  - Fix the unknown column error for SQL statements like `SELECT a FROM t HAVING t.a` #18432
  - Forbid adding a primary key for a table when the table has no primary key or when the table already has an integer primary key #18342
  - Return an empty set when executing `EXPLAIN FORMAT="dot" FOR CONNECTION` #17157
  - Fix `STR_TO_DATE`'s handling for format token'%r', '%h' #18725
- TiKV
  - Fix a bug that might read stale data during region merging #8111
  - Fix the issue of memory leak during the scheduling process #8355
- Tools
  - TiDB Lightning
    * Fix the issue that the `log-file` flag is ignored #345

9.2.5 TiDB 3.0.16 Release Notes

Release date: July 03, 2020

TiDB version: 3.0.16

9.2.5.1 Improvements

- TiDB
  - Support the `is null` filter condition in hash partition pruning #17308
  - Assign different `Backoffers` to each Region to avoid the SQL timeout issue when multiple Region requests fail at the same time #17583
  - Split separate Regions for the newly added partition #17668
  - Discard feedbacks generated from the `delete` or `update` statement #17841
  - Correct the usage of `json.Unmarshal` in `job.DecodeArgs` to be compatible with future Go versions #17887
  - Remove sensitive information in the slow query log and the statement summary table #18128
  - Match the MySQL behavior with `DateTime` delimiters #17499
  - Handle `%h` in date formats in the range that is consistent with MySQL #17496
- TiKV
  - Avoid sending store heartbeats to PD after snapshots are received #8145
  - Improve the PD client log #8091

9.2.5.2 Bug Fixes

- TiDB
  - Fix the data inconsistency issue occurred because the lock of a written and deleted primary key in one transaction is resolved by another transaction #18248
  - Fix the `Got too many pings` gRPC error log in the PD server-side followers 17944
  - Fix the panic issue that might occur when the child of HashJoin returns the `TypeNull` column #17935
  - Fix the error message when access is denied #17722
  - Fix JSON comparison issue for the `int` and `float` types #17715
  - Update the failpoint which causes data race #17710
  - Fix the issue that the timeout pre-split Regions might not work when creating tables #17617
  - Fix the panic caused by ambiguous error messages after the sending failure #17378
  - Fix the issue that `FLASHBACK TABLE` might fail in some special cases #17165

  - Fix the issue of inaccurate range calculation results when statements only have string columns #16658
  - Fix the query error occurred when the `only_full_group_by` SQL mode is set #16620
  - Fix the issue that the field length of results returned from the `case when` function is inaccurate #16562
  - Fix the type inference for the decimal property in the `count` aggregate function #17702

- TiKV

  - Fix the potential wrong result read from ingested files #8039
  - Fix the issue that a peer can not be removed when its store is isolated during multiple merge processes #8005

- PD

  - Fix the `404` error when querying Region keys in PD Control #2577

9.2.6 TiDB 3.0.15 Release Notes

Release date: June 5, 2020

TiDB version: 3.0.15

9.2.6.1 New Features

- TiDB

  - Forbid the query in partitioned tables to use the plan cache feature #16759
  - Support the `admin recover index` and `admin check index` statements on partitioned tables #17315 #17390
  - Support partition pruning of the `in` condition for Range partitioned tables #17318
  - Optimize the output of `SHOW CREATE TABLE`, and add quotation marks to the partition name #16315
  - Support the `ORDER BY` clause in the `GROUP_CONCAT` function #16988
  - Optimize the memory allocation mechanism of `CMSketch` statistics to reduce the impact of garbage collection (GC) on performance #17543

- PD

  - Add a policy in which PD performs scheduling in terms of the number of Leaders #2479

9.2.6.2 Bug Fixes

- TiDB
  - Use deep copy to copy the `enum` and `set` type data in the `Hash` aggregate function; fix an issue of correctness #16890
  - Fix the issue that `PointGet` returns incorrect results because of the wrong processing logic of integer overflow #16753
  - Fix the issue of incorrect results caused by incorrect processing logic when the `CHAR()` function is used in the query predicate #16557
  - Fix the issue of inconsistent results in the storage layer and calculation layer of the `IsTrue` and `IsFalse` functions #16627
  - Fix the incorrect `NotNull` flags in some expressions, such as `case when` #16993
  - Fix the issue that the optimizer cannot find a physical plan for `TableDual` in some scenarios #17014
  - Fix the issue that the syntax for partition selection does not take effect correctly in the Hash partitioned table #17051
  - Fix the inconsistent results between TiDB and MySQL when XOR operates on a floating-point number #16976
  - Fix the error that occurs when executing DDL statement in the prepared manner #17415
  - Fix the incorrect processing logic of computing the batch size in the ID allocator #17548
  - Fix the issue that the `MAX_EXEC_TIME` SQL hint does not take effect when the time exceeds the expensive threshold #17534
- TiKV
  - Fix the issue that memory defragmentation is not effective after running for a long time #7790
  - Fix the panic issue caused by incorrectly removing snapshot files after TiKV is restarted accidentally #7925
  - Fix the gRPC disconnection caused by too large message packages #7822

9.2.7 TiDB 3.0.14 Release Notes

Release date: May 9, 2020

TiDB version: 3.0.14

9.2.7.1 Compatibility Changes

- TiDB
  - Adjust the user privilege in `performance_schema` and `metrics_schema` from read-write to read-only #15417

9.2.7.2 Important Bug Fixes

- TiDB
  - Fix the issue that the query result of `index join` is incorrect when the `join` condition has multiple equivalent conditions on the column with the `handle` attribute #15734
  - Fix the panic that occurs when performing the `fast analyze` operation on the column with the `handle` attribute #16079
  - Fix the issue that the `query` field in the DDL job structure is incorrect when the DDL statement is executed in a way of `prepare`. This issue might cause data inconsistency between the upstream and the downstream when Binlog is used for data replication. #15443
- TiKV
  - Fix the issue that repeated requests on the cleanup of lock might destroy the atomicity of the transaction #7388

9.2.7.3 New Features

- TiDB
  - Add the schema name column and the table name column to the query results of the `admin show ddl jobs` statement #16428
  - Enhance the `RECOVER TABLE` syntax to support recovering truncated tables #15458
  - Support the privilege check for the `SHOW GRANTS` statement #16168
  - Support the privilege check for the `LOAD DATA` statement #16736
  - Improve the performance of partition pruning when functions related to time and date are used as partition keys #15618
  - Adjust the log level of `dispatch error` from `WARN` to `ERROR` #16232
  - Support the `require-secure-transport` startup option to force clients to use TLS #15415
  - Support HTTP communication between TiDB components when TLS is configured #15419
  - Add the `start_ts` information of the current transaction to the `information_schema.processlist` table #16160
  - Support automatically reloading the TLS certificate information used for communication among clusters #15162
  - Improve the read performance of the partitioned tables by restructuring the partition pruning #15628
  - Support the partition pruning feature when `floor(unix_timestamp(a))` is used as the partition expression of the `range` partition table #16521

  - Allow executing the `update` statement that contains a `view` and does not update the `view` #16787
  - Prohibit creating nested `views` #15424
  - Prohibit truncating `view` #16420
  - Prohibit using the `update` statement to explicitly update the values of a column when this column is not in the `public` state #15576
  - Prohibit starting TiDB when the `status` port is occupied #15466
  - Change the character set of the `current_role` function from `binary` to `utf8mb4` #16083
  - Improve `max-execution-time` usability by checking the interrupt signal when the data of a new Region is read #15615
  - Add the `ALTER TABLE ... AUTO_ID_CACHE` syntax for explicitly setting the cache step of `auto_id` #16287

- TiKV

  - Improve the performance when many conflicts and the `BatchRollback` condition exist in optimistic transactions #7605
  - Fix the issue of decreased performance that occurs because the pessimistic lock `waiter` is frequently awakened when many conflicts exist in pessimistic transactions #7584

- Tools

  - TiDB Lightning

    * Support printing the TiKV cluster mode using the `fetch-mode` sub-command of tidb-lightning-ctl #287

9.2.7.4 Bug Fixes

- TiDB

  - Fix the issue that `WEEKEND` function is not compatible with MySQL when the SQL mode is `ALLOW_INVALID_DATES` #16170
  - Fix the issue that the `DROP INDEX` statement fails to execute when the index column contains the auto-increment primary key #16008
  - Fix the issue of incorrect values of the `TABLE_NAMES` column in the Statement Summary #15231
  - Fix the issue that some expressions have incorrect results when the plan cache is enabled #16184
  - Fix the issue that the result of the `not`/`istrue`/`isfalse` function is incorrect #15916
  - Fix the panic caused by the `MergeJoin` operation on tables with redundant indexes #15919

- Fix the issue caused by incorrectly simplifying the link when the predicate only refers to the outer table #16492
- Fix the issue that the `CURRENT_ROLE` function reports an error caused by the `SET ROLE` statement #15569
- Fix the issue that the result of the `LOAD DATA` statement is incompatible with MySQL when this statement encounters `\` #16633
- Fix the issue that the database visibility is incompatible with MySQL #14939
- Fix the issue of incorrect privilege check for the `SET DEFAULT ROLE ALL` statement #15585
- Fix the issue of partition pruning failure caused by the plan cache #15818
- Fix the issue that `schema change` is reported during the transaction commit when concurrent DDL operations are performed on a table and blocking exists, because the transaction does not lock the related table #15707
- Fix the incorrect behavior of `IF(not_int, *, *)` #15356
- Fix the incorrect behavior of `CASE WHEN (not_int)` #15359
- Fix the issue that the `Unknown column` error message is returned when using a `view` that is not in the current schema #15866
- Fix the issue that the result of parsing time strings is incompatible with MySQL #16242
- Fix the possible panic of the collation operator in `left join` when a `null` column exists in the right child node #16528
- Fix the issue that no error message is returned even though the SQL execution is blocked when TiKV keeps returning the `StaleCommand` error message #16528
- Fix the possible panic caused by the port probing when the audit plugin is enabled #15967
- Fix the panic caused when `fast analyze` works on indices only #15967
- Fix the possible panic of the `SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST` statement execution in some cases #16309
- Fix the issue of TiDB OOM caused by specifying a large number of partitions (for example, `9999999999999`) when the hash partition table is created without checking the number of partitions before allocating memory #16218
- Fix the issue of incorrect information of partitioned tables in `information_schema` ↪ `.tidb_hot_table` #16726
- Fix the issue that the partition selection algorithm does not take effect on the hash partitioned table #16070
- Fix the issue that the HTTP API of the MVCC series does not support partitioned tables #16191
- Keep the error handling of the `UNION` statement consistent with that of the `SELECT` statement #16137
- Fix the issue of incorrect behavior when the parameter type of the `VALUES` function is `bit(n)` #15486
- Fix the issue that the processing logic of TiDB is inconsistent with MySQL when the `view` column name is too long. In this case, the system automatically generates a short column name. #14873
- Fix the issue that `(not not col)` is incorrectly optimized as `col` #16094

- Fix the issue of incorrect `range` of the inner table built by `IndexLookupJoin` plans #15753
- Fix the issue that `only_full_group_by` fails to correctly check expressions with brackets #16012
- Fix the issue that an error is returned when the `select view_name.col_name from view_name` statement is executed #15572

- TiKV
  - Fix the issue that the node cannot be deleted correctly after the isolation recovery in some cases #7703
  - Fix the issue of data loss during network isolation caused by the Region Merge operation #7679
  - Fix the issue that learner cannot be removed correctly in some cases #7598
  - Fix the issue that the scanning result of raw key-value pairs might be out of order #7597
  - Fix the issue of reconnection when the batch of Raft messages is too large #7542
  - Fix the issue of gRPC thread deadlock caused by the empty request #7538
  - Fix the issue that the processing logic of restarting the learner is incorrect during the merge process #7457
  - Fix the issue that repeated requests on the cleanup of lock might destroy the atomicity of the transaction #7388

9.2.8 TiDB 3.0.13 Release Notes

Release date: April 22, 2020

TiDB version: 3.0.13

9.2.8.1 Bug Fixes

- TiDB
  - Fix the issue caused by unchecked `MemBuffer` that the `INSERT ... ON DUPLICATE KEY UPDATE` statement might be executed incorrectly within a transaction when users need to insert multiple rows of duplicate data #16690

- TiKV
  - Fix the issue that the system might get stuck and the service is unavailable if `Region Merge` is executed repeatedly #7612

9.2.9 TiDB 3.0.12 Release Notes

Release date: March 16, 2020

TiDB version: 3.0.12

TiDB Ansible version: 3.0.12

> **Warning:**
>
> Some known issues are found in this version, and these issues are fixed in new versions. It is recommended that you use the latest 3.0.x version.

9.2.9.1 Compatibility Changes

- TiDB
  - Fix the issue of inaccurate timing of prewrite binlog in slow query log. The original timing field was called `Binlog_prewrite_time`. After this fix, the name is changed to `Wait_prewrite_binlog_time`. #15276

9.2.9.2 New Features

- TiDB
  - Support dynamic loading of the replaced certificate file by using the `alter instance` statement #15080 #15292
  - Add the `cluster-verify-cn` configuration item. After configuration, the status service can only be used when with the corresponding CN certificate. #15164
  - Add a flow limiting feature for DDL requests in each TiDB server to reduce the error reporting frequency of DDL request conflicts #15148
  - Support exiting of the TiDB server when binlog write fails #15339
- Tools
  - TiDB Binlog
    * Add the `kafka-client-id` configuration item in Drainer, which supports connecting to Kafka clients to configure the client ID #929

9.2.9.3 Bug Fixes

- TiDB
  - Make `GRANT`, `REVOKE` guarantee atomicity when modifying multiple users #15092

  - Fix the issue that the locking of pessimistic lock on the partition table failed to lock the correct row #15114
  - Make the error message display according to the value of `max-index-length` in the configuration when the index length exceeds the limit #15130
  - Fix the incorrect decimal point issue of the `FROM_UNIXTIME` function #15270
  - Fix the issue of conflict detection failure or data index inconsistency caused by deleting records written by oneself in a transaction #15176
- TiKV
  - Fix the issue of conflict detection failure or data index inconsistency caused by inserting an existing key into a transaction and then deleting it immediately when disabling the consistency check parameter #7054
  - Introduce a flow control mechanism in Raftstore to solve the problem that without flow control, it might lead to too slow tracking and cause the cluster to be stuck, and the transaction size might cause frequent reconnection of TiKV connections #7072 #6993
- PD
  - Fix the issue of incorrect Region information caused by data race when PD processes Region heartbeats #2233
- TiDB Ansible
  - Support deploying multiple Grafana/Prometheus/Alertmanager in a cluster #1198

9.2.10 TiDB 3.0.11 Release Notes

Release date: March 4, 2020

TiDB version: 3.0.11

TiDB Ansible version: 3.0.11

> **Warning:**
>
> Some known issues are found in this version, and these issues are fixed in new versions. It is recommended that you use the latest 3.0.x version.

9.2.10.1 Compatibility Changes

- TiDB
  - Add the `max-index-length` configuration item to control the maximum index length, which is compatible with the behavior of TiDB versions before 3.0.7 or of MySQL #15057

9.2.10.2 New Features

- TiDB
  - Support showing the meta information of partitioned tables in the `information_schema.PARTITIONS` table #14849
- TiDB Binlog
  - Support the bidirectional data replication between TiDB clusters #884 #909
- TiDB Lightning
  - Support the TLS configuration #44 #270
- TiDB Ansible
  - Modify the logic of `create_users.yml` so that users of the central control machine do not have to be consistent with `ansible_user` #1184

9.2.10.3 Bug Fixes

- TiDB
  - Fix the issue of Goroutine leaks when retrying an optimistic transaction because queries using `Union` are not marked read-only #15076
  - Fix the issue that `SHOW TABLE STATUS` fails to correctly output the table status at the snapshot time because the value of the `tidb_snapshot` parameter is not correctly used when executing the `SET SESSION tidb_snapshot = 'xxx';` statement #14391
  - Fix the incorrect result caused by a SQL statement that contains `Sort Merge Join` and `ORDER BY DESC` at the same time #14664
  - Fix the panic of TiDB server when creating partition tables using the unsupported expression. The error information `This partition function is not allowed` is returned after fixing this panic. #14769
  - Fix the incorrect result occurred when executing the `select max()from subquery` statement with the subquery containing `Union` #14944
  - Fix the issue that an error message is returned when executing the `SHOW BINDINGS` statement after executing `DROP BINDING` that drops the execution binding #14865
  - Fix the issue that the connection is broken because the maximum length of an alias in a query is 256 characters in the MySQL protocol, but TiDB does not cut the alias in the query results according to this protocol #14940
  - Fix the incorrect query result that might occur when using the string type in `DIV`. For instance, now you can correctly execute the `select 1 / '2007' div 1` statement #14098
- TiKV
  - Optimize the log output by removing unnecessary logs #6657

  - Fix the panic that might occur when the peer is removed under high loads #6704
  - Fix the issue that Hibernate Regions are not waken up in some cases #6732 #6738

- TiDB Ansible

  - Update outdated document links in `tidb-ansible` #1169
  - Fix the issue that undefined variables might occur in the `wait for region replication complete` task #1173

9.2.11 TiDB 3.0.10 Release Notes

Release date: February 20, 2020

TiDB version: 3.0.10

TiDB Ansible version: 3.0.10

> **Warning:**
>
> Some known issues are found in this version, and these issues are fixed in new versions. It is recommended that you use the latest 3.0.x version.

9.2.11.1 TiDB

- Fix wrong `Join` results when `IndexLookUpJoin` uses `OtherCondition` to construct `InnerRange` #14599
- Delete the `tidb_pprof_sql_cpu` configuration item and add the `tidb_pprof_sql_cpu` variable #14416
- Fix the issue that users can query all databases only when they have global privileges #14386
- Fix the issue that data visibility does not meet expectations due to transaction timeout when executing `PointGet` operations #14480
- Change the timing of pessimistic transaction activation to delayed activation, consistent with the optimistic transaction mode #14474
- Fix the incorrect time zone results when the `unixtimestamp` expression calculates the time zone of the table partitions #14476
- Add the `tidb_session_statement_deadlock_detect_duration_seconds` monitoring item to monitor deadlock detection duration #14484
- Fix the system panic issue caused by some logic errors of GC workers #14439
- Correct the expression name of the `IsTrue` function #14516
- Fix the issue that some memory usage is counted inaccurately #14533
- Fix the system panic issue caused by incorrect processing logic during CM-Sketch statistics initialization #14470

- Fix the issue of inaccurate partition pruning when querying partitioned tables #14546
- Fix the issue that the default database name of the SQL statement in SQL bindings is set incorrectly #14548
- Fix the issue that `json_key` is not compatible with MySQL #14561
- Add the feature of automatically updating the statistics of partitioned tables #14566
- Fix the issue that the plan ID changes when the `PointGet` operation is executed (the plan ID is expected to be `1` always) #14595
- Fix the system panic issue caused by incorrect processing logic when SQL bindings do not match exactly #14263
- Add the `tidb_session_statement_pessimistic_retry_count` monitoring item to monitor the number of retries after the failure to lock pessimistic transactions #14619
- Fix the incorrect privilege check for `show binding` statements #14618
- Fix the issue that the query cannot be killed because the `backoff` logic does not include checking the `killed` tag #14614
- Improve the performance of statement summary by reducing the time to hold internal locks #14627
- Fix the issue that TiDB's result of parsing strings to time is incompatible with MySQL #14570
- Record the user login failures in audit logs #14620
- Add the `tidb_session_ statement_lock_keys_count` monitoring item to monitor the number of lock keys for pessimistic transactions #14634
- Fix the issue that characters in JSON such as `&`, `<`, and `>` are incorrectly escaped #14637
- Fix the system panic issue caused by excessive memory usage when the `HashJoin` operation is building a hash table #14642
- Fix the panic issue caused by incorrect processing logic when an SQL binding processes illegal records #14645
- ix a MySQL incompatibility issue by adding Truncated error detection to decimal division calculation #14673
- Fix the issue of successfully granting users privileges on a table that does not exist #14611

9.2.11.2 TiKV

- Raftstore
  - Fix the system panic issue #6460 or data loss issue #598 caused by Region merge failure #6481
  - Support `yield` to optimize scheduling fairness, and support pre-transfering the leader to improve leader scheduling stability #6563

9.2.11.3 PD

- Fix the invalid cache issue by supporting automatically updating the Region cache information when the system traffic changes #2103

- Use leader lease time to determine TSO service validity #2117

9.2.11.4 Tools

- TiDB Binlog
  - Support relay log in Drainer #893
- TiDB Lightning
  - Make some configuration items use default values when a config file is missing #255
  - Fix the issue that the web interface cannot be opened in the non-server mode #259

9.2.11.5 TiDB Ansible

- Fix the issue that the command execution fails due to the failure to obtain PD leader in some scenarios #1121
- Add the `Deadlock Detect Duration` monitoring item in the TiDB dashboard #1127
- Add the `Statement Lock Keys Count` monitoring item in the TiDB dashboard #1132
- Add the `Statement Pessimistic Retry Count` monitoring item in the TiDB dashboard #1133

9.2.12 TiDB 3.0.9 Release Notes

Release date: January 14, 2020

TiDB version: 3.0.9

TiDB Ansible version: 3.0.9

> **Warning:**
>
> Some known issues are found in this version, and these issues are fixed in new versions. It is recommended that you use the latest 3.0.x version.

9.2.12.1 TiDB

- Executor
  - Fix the incorrect result when the aggregate function is applied to the `ENUM` column and the collection column #14364

- Server
  - Support the `auto_increment_increment` and `auto_increment_offset` system variables #14396
  - Add the `tidb_tikvclient_ttl_lifetime_reach_total` monitoring metric to monitor the number of pessimistic transactions with a TTL of 10 minutes #14300
  - Output the SQL information in the log when the SQL query causes a panic during its execution #14322
  - Add the `plan` and `plan_digest` fields in the statement summary table to record the `plan` that is being executed and the `plan` signature #14285
  - Adjust the default value of the `stmt-summary.max-stmt-count` configuration item from `100` to `200` #14285
  - Add the `plan_digest` field in the slow query table to record the `plan` signature #14292
- DDL
  - Fix the issue that the results of anonymous indexes created using `alter table ... add index` on the `primary` column is inconsistent with MySQL #14310
  - Fix the issue that `VIEW`s are mistakenly dropped by the `drop table` syntax #14052
- Planner
  - Optimize the performance of statements such as `select max(a), min(a) from t`. If an index exists in the `a` column, the statement is optimized to `select * from (select a from t order by a desc limit 1)as t1, (select a from t order by a limit 1)as t2` to avoid full table scan #14410

9.2.12.2 TiKV

- Raftstore
  - Speed up the configuration change to speed up the Region scattering #6421
- Transaction
  - Add the `tikv_lock_manager_waiter_lifetime_duration`, `tikv_lock_manager_detect_dura` , and `tikv_lock_manager_detect_duration` monitoring metrics to monitor `waiters`' lifetime, the time cost of detecting deadlocks, and the status of `Wait` table #6392
  - Optimize the following configuration items to reduce transaction execution latency caused by changing Region leader or the leader of deadlock detector in extreme situations #6429
    * Change the default value of `wait-for-lock-time` from `3s` to `1s`
    * Change the default value of `wake-up-delay-duration` from `100ms` to `20ms`
  - Fix the issue that the leader of the deadlock detector might be incorrect during the Region Merge process #6431

9.2.12.3 PD

- Support using backlash `/` in the location label name #2083
- Fix the incorrect statistics because the tombstone store is mistakenly included by the label counter #2067

9.2.12.4 Tools

- TiDB Binlog
  - Add the unique key information in the binlog protocol output by Drainer #862
  - Support using the encrypted password for database connection for Drainer #868

9.2.12.5 TiDB Ansible

- Support automatically creating directories to optimize the deployment of TiDB Lightning #1105

9.2.13 TiDB 3.0.8 Release Notes

Release date: December 31, 2019

TiDB version: 3.0.8

TiDB Ansible version: 3.0.8

9.2.13.1 TiDB

- SQL Optimizer
  - Fix the wrong SQL binding plan caused by untimely cache updates #13891
  - Fix the issue that the SQL binding might be invalid when an SQL statement contains a symbol list #14004
  - Fix the issue that an SQL binding cannot be created or deleted because an SQL statement ends with `;` #14113
  - Fix the issue that a wrong SQL query plan might be selected because the `PhysicalUnionScan` operator sets wrong statistics #14133
  - Remove the `minAutoAnalyzeRatio` restriction to make `autoAnalyze` more timely #14015
- SQL Execution Engine
  - Fix issues that the `INSERT/REPLACE/UPDATE ... SET ... = DEFAULT` syntax might report an error and combining the usage of the `DEFAULT` expression with a virtual generated column might report an error #13682

  - Fix the issue that the `INSERT` statement might report an error when converting a string to a float #14011
  - Fix the issue that sometimes the aggregate operation is low effective because the concurrency value of the `HashAgg` executor is incorrectly initialized #13811
  - Fix the issue that an error is reported in the execution of`group by item` when the clause is in the parentheses #13658
  - Fix the issue that the execution of `OUTER JOIN` might report an error because TiDB incorrectly calculates `group by item` #14014
  - Fix the issue that the error message is inaccurate when Range-exceeding data is written into Range partitioned tables #14107
  - Revert PR #10124 and cancel the `PadCharToFullLength` effect to avoid unexpected query results in special cases, considering that MySQL 8 will discard `PadCharToFullLength` soon #14157
  - Fix the goroutine leak issue when executing the `EXPLAIN ANALYZE` statement caused by unguaranteed `close()` calling in `ExplainExec` #14226

- DDL

  - Optimize the error message output of `change column`/`modify column` to make it easier to understand #13796
  - Add the `SPLIT PARTITION TABLE` syntax to support splitting Regions for partitioned tables #13929
  - Fix the issue that the index length exceeds 3072 bytes and no error is reported because the index length is incorrectly checked when an index is created #13779
  - Fix the issue that the `GC life time is shorter than transaction duration` error message might be reported because it takes too much time to add an index in partitioned tables #14132
  - Fix the panic when `SELECT * FROM information_schema.KEY_COLUMN_USAGE` is executed because the foreign key is not checked when `DROP COLUMN`/`MODIFY COLUMN`/`CHANGE COLUMN` is executed #14105

- Server

  - Statement Summary improvements:
    * Add a large number of SQL metric fields to facilitate analyzing SQL statements in more detail #14151, #14168
    * Add the `stmt-summary.refresh-interval` parameter to control whether to move the stale data from the `events_statements_summary_by_digest` table to the `events_statements_summary_by_digest_history` table (the default interval: 30 minutes) #14161
    * Add the `events_statements_summary_by_digest_history` table to save the stale data in `events_statements_summary_by_digest` #14166
  - Fix the issue that the binlog is incorrectly output when RBAC-related internal SQL statements are executed #13890
  - Add the `server-version` configuration item to control the feature of modifying the TiDB server version #13906

- Add the feature of using the HTTP interface to recover writing the TiDB binlog #13892
- Update the privilege required by `GRANT roles TO user` from `GrantPriv` to `ROLE_ADMIN` or `SUPER`, to keep consistency with the MySQL behavior #13932
- Modify the TiDB behavior from using the current database to reporting the `No database selected` error when the `GRANT` statement does not specify a database name, to keep compatibility with the MySQL behavior #13784
- Modify the execution privilege for the `REVOKE` statement from `SuperPriv` to `REVOKE` being executable only if the user has the privilege for the corresponding schema, to keep consistency with the MySQL behavior #13306
- Fix the issue that `GrantPriv` is mistakenly granted to the target user when the `GRANT ALL` syntax does not contain `WITH GRANT OPTION` #13943
- Fix the issue that the error message does not contain the cause for the `LOAD DATA` statement's wrong behavior when `LoadDataInfo` fails to call `addRecord` #13980
- Fix the issue that wrong slow query information is output because multiple SQL statements in a query share the same `StartTime` #13898
- Fix the issue that the memory might leak when `batchClient` processes a large transaction #14032
- Fix the issue that `system_time_zone` is always displayed as `CST` and now TiDB's `system_time_zone` is obtained from `systemTZ` in the `mysql.tidb` table #14086
- Fix the issue that the `GRANT ALL` syntax does not grant all privileges to the user #14092
- Fix the issue that the `Priv_create_user` privilege is invalid for `CREATE ROLE` and `DROP ROLE` #14088
- Modify the error code of `ErrInvalidFieldSize` from `1105(Unknow Error)` to `3013` #13737
- Add the `SHUTDOWN` command to stop a TiDB server and add the `ShutdownPriv` privilege #14104
- Fix the atomicity issue for the `DROP ROLE` statement to avoid some roles being deleted unexpectedly when TiDB fails to execute a statement #14130
- Fix the issue that the `tidb_enable_window_function` in the `SHOW VARIABLE` result incorrectly outputs `1` when a TiDB version is upgraded to 3.0, and replace the wrong result with `0` #14131
- Fix the issue that the goroutine might leak because `gcworker` continuously retries when the TiKV node is offline #14106
- Record the binlog `Prewrite` time in the slow query log to improve the usability for issue tracking #14138
- Make the `tidb_enable_table_partition` variable support `GLOBAL SCOPE` #14091
- Fix the issue that the user privilege might be missing or mistakenly added because the newly added privilege is not correctly granted to the corresponding user when a new privilege is added #14178
- Fix the issue that the `CheckStreamTimeoutLoop` goroutine might leak because `rpcClient` does not close when the TiKV server is disconnected #14227
- Support certificate-based authentication (User document) #13955

- Transaction
  - Update the default value of the `tidb_txn_mode` variable from `""` to `"pessimistic "` when a new cluster is created #14171
  - Fix the issue that the lock waiting time is too long for a pessimistic transaction because the lock waiting time for a single statement is not reset when a transaction is retried #13990
  - Fix the issue that wrong data might be read because unmodified data is unlocked for the pessimistic transaction model #14050
  - Fix repeated insert value restriction checks because transaction types are not distinguished when prewrite is performed in mocktikv #14175
  - Fix the panic because transactions are not correctly handled when `session.TxnState` is `Invalid` #13988
  - Fix the issue that the `ErrConfclit` structure in mocktikv does not contain `ConflictCommitTS` #14080
  - Fix the issue that the transaction is blocked because TiDB does not correctly check lock timeout after resolving the lock #14083
- Monitor
  - Add the `pessimistic_lock_keys_duration` monitoring item in `LockKeys` #14194

9.2.13.2 TiKV

- Coprocessor
  - Modify the level of the output log from `error` to `warn` when an error occurs in Coprocessor #6051
  - Modify the update behavior of statistics sampling data from directly updating the row to deleting before inserting, to keep consistency with the update behavior of tidb-server #6069
- Raftstore
  - Fix the panic caused by repeatedly sending the `destroy` message to `peerfsm` and `peerfsm` being destroyed multiple times #6297
  - Update the default value of `split-region-on-table` from `true` to `false` to disable splitting Regions by table by default #6253
- Engine
  - Fix the issue that empty data might be returned because RocksDB iterator errors are not correctly processed in extreme conditions #6326
- Transaction
  - Fix the issue that TiKV fails to write data into keys and GC is blocked because the pessimistic locks are incorrectly cleaned up #6354

  - Optimize the pessimistic lock waiting mechanism to improve the performance in scenarios where the lock conflict is severe #6296

- Update the default value of `tikv_alloc` from `tikv_alloc/default` to `jemalloc` #6206

9.2.13.3 PD

- Client
  - Support using `context` to create a client and setting the timeout duration when creating a new client #1994
  - Support creating the `KeepAlive` connection #2035
- Optimize the performance for the `/api/v1/regions` API #1986
- Fix the issue that deleting stores in a `tombstone` state might cause a panic #2038
- Fix the issue that overlapped Regions are mistakenly deleted when loading the Region information from disks #2011, #2040
- Upgrade etcd from v3.4.0 to v3.4.3 (note that after upgrading you can only degrade etcd using pd-recover) #2058

9.2.13.4 Tools

- TiDB Binlog
  - Fix the issue that the binlog is ignored because Pump does not receive the DDL committed binlog #853

9.2.13.5 TiDB Ansible

- Revert the simplified configuration item #1053
- Optimize the logic for checking the TiDB version when performing a rolling update #1056
- Upgrade TiSpark to v2.1.8 #1061
- Fix the issue that the PD role monitoring item is wrongly displayed on Grafana #1065
- Optimize `Thread Voluntary Context Switches` and `Thread Nonvoluntary Context Switches` monitoring items on the TiKV Detail page on Grafana #1071

9.2.14 TiDB 3.0.7 Release Notes

Release date: December 4, 2019

TiDB version: 3.0.7

TiDB Ansible version: 3.0.7

9.2.14.1 TiDB

- Fix the issue that the lock TTL's value is too large because the TiDB server's local time is behind PD's timestamp #13868
- Fix the issue that the timezone is incorrect after parsing the date from strings using `gotime.Local` #13793
- Fix the issue that the result might be incorrect because the `binSearch` function does not return an error in the implementation of `builtinIntervalRealSig` #13767
- Fix the issue that data is incorrect because the precision is lost when an integer is converted to an unsigned floating point or decimal type #13755
- Fix the issue that the result is incorrect because the `not null` flag is not properly reset when the `USING` clause is used in Natural Outer Join and Outer Join #13739
- Fix the issue that the statistics are not accurate because a data race occurs when statistics are updated #13687

9.2.14.2 TiKV

- Make the deadlock detector only observe valid Regions to make sure the deadlock manager is in a valid Region #6110
- Fix a potential memory leak issue #6128

9.2.15 TiDB 3.0.6 Release Notes

Release date: November 28, 2019

TiDB version: 3.0.6

TiDB Ansible version: 3.0.6

9.2.15.1 TiDB

- SQL Optimizer
  - Fix the issue that the result is incorrect after the window function AST restores SQL text, for example, `over w` being mistakenly restored to `over (w)` #12933
  - Fix the issue of pushing down `STREAM AGG()` to `doubleRead` #12690
  - Fix the issue that quotes are incorrectly handled for SQL binding #13117
  - Optimize the `select max(_tidb_rowid)from t` scenario to avoid full table scans #13095
  - Fix the issue that the query result is incorrect when the query statement contains a variable assignment expression #13231
  - Fix the issue that the result is incorrect when the `UPDATE` statement contains both a sub-query and a generated column; fix the `UPDATE` statement execution error when this statement contains two same-named tables from different source databases #13350

  - Support `_tidb_rowid` for point queries #13416
  - Fix the issue that the generated query execution plan is incorrect, caused by incorrect usage of partitioned table statistics #13628

- SQL Execution Engine

  - Fix the issue that TiDB is incompatible with MySQL when handling invalid values of the year type #12745
  - Reuse `Chunk` in the `INSERT ON DUPLICATE UPDATE` statement to reduce the memory overhead #12998
  - Add the support for the `JSON_VALID` built-in function #13133
  - Support executing `ADMIN CHECK TABLE` on partitioned tables #13140
  - Fix the panic issue when `FAST ANALYZE` is executed on empty tables #13343
  - Fix the panic issue when executing `FAST ANALYZE` on an empty table that contains multi-column indexes #13394
  - Fix the issue that the estimated number of rows is greater than 1 when the `WHERE` clause contains an equal condition on the unique key #13382
  - Fix the issue that the returned data might be duplicated when `Streaming` is enabled in TiDB #13254
  - Extract the top N values from the count-min sketch to improve the estimation accuracy #13429

- Server

  - Make requests sent to TiKV fail quickly when the gRPC dial times out #12926
  - Add the following virtual tables: #13009
    * `performance_schema.tidb_profile_allocs`
    * `performance_schema.tidb_profile_block`
    * `performance_schema.tidb_profile_cpu`
    * `performance_schema.tidb_profile_goroutines`
  - Fix the issue that the `kill` command does not work when the query is waiting for pessimistic locking #12989
  - Do not do asynchronous rollback when acquiring pessimistic locking fails and the transaction only involves modifying a single key #12707
  - Fix the panic issue when the response for the request of splitting Regions is empty #13092
  - Avoid unnecessary backoff when `PessimisticLock` returns a locking error #13116
  - Modify the TiDB behavior for checking configurations by printing a warning log for unrecognized configuration option #13272
  - Support obtaining the binlog status of all TiDB nodes via the `/info/all` interface #13187
  - Fix the issue that goroutine might leak when TiDB kills connections #13251
  - Make the `innodb_lock_wait_timeout` parameter work in pessimistic transactions to control the lock wait timeout for pessimistic locking #13165
  - Stop updating pessimistic transaction TTL when pessimistic transactional queries are killed to prevent other transactions from waiting unnecessarily #13046

- DDL
  - Fix the issue that the execution result of `SHOW CREATE VIEW` in TiDB is inconsistent with that in MySQL #12912
  - Support creating `View` based on `union`, for example, `create view v as select * from t1 union select * from t2` #12955
  - Add more transaction-related fields for the `slow_query` table: #13072
    - `Prewrite_time`
    - `Commit_time`
    - `Get_commit_ts_time`
    - `Commit_backoff_time`
    - `Backoff_types`
    - `Resolve_lock_time`
    - `Local_latch_wait_time`
    - `Write_key`
    - `Write_size`
    - `Prewrite_region`
    - `Txn_retry`
  - Use the table's `COLLATE` instead of the system's default charset in the column when a table is created and the table contains `COLLATE` #13174
  - Limit the length of the index name when creating a table #13310
  - Fix the issue that the table name length is not checked when a table is renamed #13346
  - Add the `alter-primary-key` configuration (disabled by default) to support adding/dropping the primary key in TiDB #13522

9.2.15.2 TiKV

- Fix the issue that the `acquire_pessimistic_lock` interface returns a wrong `txn_size` #5740
- Limit the writes for GC worker per second to reduce the impact on the performance #5735
- Make `lock_manager` accurate #5845
- Support `innodb_lock_wait_timeout` for pessimistic locking #5848
- Add the configuration check for Titan #5720
- Support using tikv-ctl to dynamically modify the GC I/O limit: `tikv-ctl --host=ip:port modify-tikv-config -m server -n gc.max_write_bytes_per_sec -v 10MB` #5957
- Reduce useless `clean up` requests to decrease the pressure on the deadlock detector #5965
- Avoid reducing TTL in pessimistic locking prewrite requests #6056
- Fix the issue that a missing blob file might occur in Titan #5968
- Fix the issue that `RocksDBOptions` might not take effect in Titan #6009

9.2.15.3 PD

- Add an `ActOn` dimension for each filter to indicate that each scheduler and checker is affected by the filter, and delete two unused filters: `disconnectFilter` and `rejectLeaderFilter` #1911
- Print a warning log when it takes more than 5 milliseconds to generate a timestamp in PD #1867
- Lower the client log level when passing unavailable endpoint to the client #1856
- Fix the issue that the gRPC message package might exceed the maximum size in the `region_syncer` replication process #1952

9.2.15.4 Tools

- TiDB Binlog
  - Obtain the initial replication timestamp from PD when `initial-commit-ts` is set to "-1" in Drainer #788
  - Decouple Drainer's `Checkpoint` storage from the downstream and support saving `Checkpoint` in MySQL or local files #790
  - Fix the Drainer panic issue caused by using empty values when configuring replication database/table filtering #801
  - Fix the issue that processes get into the deadlock status instead of exiting after a panic occurs because Drainer fails to apply binlog files to the downstream #807
  - Fix the issue that Pump blocks when it exits because of gRPC's `GracefulStop` #817
  - Fix the issue that Drainer fails when it receives a binlog which misses a column during the execution of a `DROP COLUMN` statement in TiDB (v3.0.6 or later) #827
- TiDB Lightning
  - Add the `max-allowed-packet` configuration (64 M by default) for the TiDB backend #248

9.2.16 TiDB 3.0.5 Release Notes

Release date: October 25, 2019

TiDB version: 3.0.5

TiDB Ansible version: 3.0.5

9.2.16.1 TiDB

- SQL Optimizer
  - Support boundary checking on Window Functions #12404

  - Fix the issue that `IndexJoin` on the partition table returns incorrect results #12712
  - Fix the issue that the `ifnull` function on the top of the outer join `Apply` operator returns incorrect results #12694
  - Fix the issue of update failure when a subquery was included in the `where` condition of `UPDATE` #12597
  - Fix the issue that outer join was incorrectly converted to inner join when the `cast` function was included in the query conditions #12790
  - Fix incorrect expression passing in the join condition of `AntiSemiJoin` #12799
  - Fix the statistics error caused by shallow copy when initializing statistics #12817
  - Fix the issue that the `str_to_date` function in TiDB returns a different result from MySQL when the date string and the format string do not match #12725

- SQL Execution Engine

  - Fix the panic issue when the `from_unixtime` function handles null #12551
  - Fix the `invalid list index` error reported when canceling DDL jobs #12671
  - Fix the issue that arrays were out of bounds when Window Functions are used #12660
  - Improve the behavior of the `AutoIncrement` column when it is implicitly allocated, to keep it consistent with the default mode of MySQL auto-increment locking ("consecutive" lock mode): for the implicit allocation of multiple `AutoIncrement` IDs in a single-line `Insert` statement, TiDB guarantees the continuity of the allocated values. This improvement ensures that the JDBC `getGeneratedKeys()` method will get the correct results in any scenario. #12602
  - Fix the issue that the query is hanged when `HashAgg` serves as a child node of `Apply` #12766
  - Fix the issue that the `AND` and `OR` logical expressions return incorrect results when it comes to type conversion #12811

- Server

  - Implement the interface function that modifies transaction TTL to help support large transactions later #12397
  - Support extending the transaction TTL as needed (up to 10 minutes) to support pessimistic transactions #12579
  - Adjust the number of times that TiDB caches schema changes and corresponding changed table information from 100 to 1024, and support modification by using the `tidb_max_delta_schema_count` system variable #12502
  - Update the behavior of the `kvrpc.Cleanup` protocol to no longer clean locks of transactions that are not overtime #12417
  - Support logging Partition table information to the `information_schema.tables` table #12631
  - Support modifying the TTL of Region Cache by configuring `region-cache-ttl` #12683
  - Support printing the execution plan compression-encoded information in the slow log. This feature is enabled by default and can be controlled by using the `slow-`

↪ `log-plan` configuration or the `tidb_record_plan_in_slow_log` variable. In addition, the `tidb_decode_plan` function can decode the execution plan column encoded information in the slow log into execution plan information. #12808
  - Support displaying memory usage information in the `information_schema.` ↪ `processlist` table #12801
  - Fix the issue that an error and an unexpected alarm might occur when the TiKV Client judges an idle connection #12846
  - Fix the issue that the `INSERT IGNORE` statement performance is decreased because `tikvSnapshot` does not properly cache the KV results of `BatchGet()` #12872
  - Fix the issue that the TiDB response speed was relatively low because of slow connection to some KV services #12814

- DDL
  - Fix the issue that the `Create Table` operation does not correctly set the Int type default value for the Set column #12267
  - Support multiple `uniques` when creating a unique index in the `Create Table` statement #12463
  - Fix the issue that populating the default value of this column for existing rows might cause an error when adding a Bit type column using `Alter Table` #12489
  - Fix the failure of adding a partition when the Range partitioned table uses a Date or Datetime type column as the partitioning key #12815
  - Support checking the consistency of the partition type and the partition key type when creating a table or adding a partition, for the Range partitioned table with the Date or Datetime type column as the partition key #12792
  - Add a check that the Unique Key column set needs to be greater than or equal to the partitioned column set when creating a Range partitioned table #12718

- Monitor
  - Add the monitoring metrics of Commit and Rollback operations to the `Transaction OPS` dashboard #12505
  - Add the monitoring metrics of `Add Index` operation progress #12390

9.2.16.2 TiKV

- Storage
  - Add a new feature of pessimistic transactions: the transaction cleanup interface supports only cleaning up locks whose TTL is outdated #5589
  - Fix the issue that Rollback of the transaction Primary key is collapsed #5646, #5671
  - Fix the issue that under pessimistic locks, point queries might return the previous version data #5634

- Raftstore

  - Reduce message flush operations in Raftstore to improve performance and reduce CPU usage #5617
  - Optimize the cost of obtaining the Region size and estimated number of keys, to reduce heartbeat overhead and CPU usage #5620
  - Fix the issue that Raftstore prints an error log and encounters a panic when getting invalid data #5643

- Engine

  - Enable RocksDB `force_consistency_checks` to improve data safety #5662
  - Fix the issue that concurrent flush operations in Titan might cause data loss #5672
  - Update the rust-rocksdb version to avoid the issue of TiKV crash and restart caused by intra-L0 compaction #5710

9.2.16.3 PD

- Improve the precision of storage occupied by Regions #1782
- Improve the output of the `--help` command #1763
- Fix the issue that the HTTP request fails to redirect after TLS is enabled #1777
- Fix the panic issue occurred when pd-ctl uses the `store shows limit` command #1808
- Improve readability of label monitoring metrics and reset the original leader's monitoring data when the leader switches, to avoid false reports #1815

9.2.16.4 Tools

- TiDB Binlog

  - Fix the issue that `ALTER DATABASE` related DDL operations cause Drainer to exit abnormally #769
  - Support querying the transaction status information for Commit binlog to improve replication efficiency #757
  - Fix the issue that a Pump panic might occur when Drainer's `start_ts` is greater than Pump's largest `commit_ts` #758

- TiDB Lightning

  - Integrate the full logic import feature of Loader and support configuring the backend mode #221

9.2.16.5 TiDB Ansible

- Add the monitoring metrics of adding index speed #986
- Simplify the configuration file content and remove parameters that users do not need to configure #1043c, #998

- Fix the monitoring expression error of performance read and performance write #e90e7
- Update the monitoring display method and the alarm rules of Raftstore CPU usage #992
- Update the TiKV CPU monitoring item in the Overview monitoring dashboard to filter out the excess monitoring content #1001

9.2.17 TiDB 3.0.4 Release Notes

Release date: October 8, 2019

TiDB version: 3.0.4

TiDB Ansible version: 3.0.4

- New features
  - Add the `performance_schema.events_statements_summary_by_digest` system table to troubleshoot performance issues at the SQL level
  - Add the `WHERE` clause in TiDB's `SHOW TABLE REGIONS` syntax
  - Add the `worker-count` and `txn-batch` configuration items in Reparo to control the recovery speed
- Improvements
  - Support batch Region split command and empty split command in TiKV to improve split performance
  - Support double linked list for RocksDB in TiKV to improve performance of reverse scan
  - Add two perf tools `iosnoop` and `funcslower` in TiDB Ansible to better diagnose the cluster state
  - Optimize the output of slow query logs in TiDB by deleting redundant fields
- Changed behaviors
  - Update the default value of `txn-local-latches.enable` to `false` to disable the default behavior of checking conflicts of local transactions in TiDB
  - Add the `tidb_txn_mode` system variable of global scope in TiDB and allow using the pessimistic lock; note that TiDB still adopts the optimistic lock by default
  - Replace the `Index_ids` field in TiDB slow query logs with `Index_names` to improve the usability of slow query logs
  - Add the `split-region-max-num` parameter in the TiDB configuration file to modify the maximum number of Regions allowed in the `SPLIT TABLE` syntax
  - Return the `Out Of Memory Quota` error instead of disconnecting the link when a SQL execution exceeds the memory limit
  - Disallow dropping the `AUTO_INCREMENT` attribute of columns in TiDB to avoid misoperations. To drop this attribute, change the `tidb_allow_remove_auto_inc` system variable

- Fixed issues

  - Fix the issue that the uncommented TiDB-specific syntax `PRE_SPLIT_REGIONS` might cause errors in the downstream database during data replication
  - Fix the issue in TiDB that the slow query logs are incorrect when getting the result of `PREPARE` + `EXECUTE` by using the cursor
  - Fix the issue in PD that adjacent small Regions cannot be merged
  - Fix the issue in TiKV that file descriptor leak in idle clusters might cause TiKV processes to exit abnormally when the processes run for a long time

- Contributors

  Our thanks go to the following contributors from the community for helping this release:

  - sduzh
  - lizhenda

9.2.17.1 TiDB

- SQL Optimizer

  - Fix the issue that invalid query ranges might be resulted when splitted by feedback #12170
  - Display the returned error of the `SHOW STATS_BUCKETS` statement in hexadecimal rather than return errors when the result contains invalid Keys #12094
  - Fix the issue that when a query contains the `SLEEP` function (for example, `select 1 from (select sleep(1))t;`), column pruning causes invalid `sleep(1)` during query #11953
  - Use index scan to lower IO when a query only concerns the number of columns rather than the table data #12112
  - Do not use any index when no index is specified in `use index()` to be compatible with MySQL #12100
  - Strictly limit the number of `TopN` records in the `CMSketch` statistics to fix the issue that the `ANALYZE` statement fails because the statement count exceeds TiDB's limit on the size of a transaction #11914
  - Fix the error occurred when converting the subqueries contained in the `Update` statement #12483
  - Optimize execution performance of the `select ... limit ... offset ...` statement by pushing the Limit operator down to the `IndexLookUpReader` execution logic #12378

- SQL Execution Engine

  - Print the SQL statement in the log when the `PREPARED` statement is incorrectly executed #12191
  - Support partition pruning when the `UNIX_TIMESTAMP` function is used to implement partitioning #12169

- Fix the issue that no error is reported when `AUTO_INCREMENT` incorrectly allocates `MAX int64` and `MAX uint64` #12162
- Add the `WHERE` clause in the `SHOW TABLE ⋯ REGIONS` and `SHOW TABLE .. INDEX ⋯ REGIONS` syntaxes #12123
- Return the `Out Of Memory Quota` error instead of disconnecting the link when a SQL execution exceeds the memory limit #12127
- Fix the issue that incorrect result is returned when `JSON_UNQUOTE` function handles JSON text #11955
- Fix the issue that `LAST INSERT ID` is incorrect when assigning values to the `AUTO_INCREMENT` column in the first row (for example, `insert into t (pk, c)values (1, 2), (NULL, 3)`) #12002
- Fix the issue that the `GROUPBY` parsing rule is incorrect in the `PREPARE` statement #12351
- Fix the issue that the privilege check is incorrect in the point queries #12340
- Fix the issue that the duration by `sql_type` for the `PREPARE` statement is not shown in the monitoring record #12331
- Support using aliases for tables in the point queries (for example, `select * from t tmp where a = "aa"`) #12282
- Fix the error occurred when not handling negative values as unsigned when inserting negative numbers into BIT type columns #12423
- Fix the incorrectly rounding of time (for example, `2019-09-11 11:17:47.999999666` should be rounded to `2019-09-11 11:17:48`.) #12258
- Refine the usage of expression blocklist (for example, `<` is equivalent to `It`.) #11975
- Add the database prefix to the message of non-existing function error (for example, `[expression:1305]FUNCTION test.std_samp does not exist`) #12111

- Server
  - Add the `Prev_stmt` field in slow query logs to output the previous statement when the last statement is `COMMIT` #12180
  - Optimize the output of slow query logs by deleting redundant fields #12144
  - Update the default value of `txn-local-latches.enable` to `false` to disable the default behavior of checking conflicts of local transactions in TiDB #12095
  - Replace the `Index_ids` field in TiDB slow query logs with `Index_names` to improve the usability of slow query logs #12061
  - Add the `tidb_txn_mode` system variable of global scope in TiDB and allow using pessimistic lock #12049
  - Add the `Backoff` field in the slow query logs to record the Backoff information in the commit phase of 2PC #12335
  - Fix the issue that the slow query logs are incorrect when getting the result of `PREPARE` + `EXECUTE` by using the cursor (for example, `PREPARE stmt1FROM SELECT * FROM t WHERE a > ?; EXECUTE stmt1 USING @variable`) #12392
  - Support `tidb_enable_stmt_summary`. When this feature is enabled, TiDB counts the SQL statements and the result can be queried by using the system table

`performance_schema.events_statements_summary_by_digest` #12308
  - Adjust the level of some logs in tikv-client (for example, change the log level of `batchRecvLoop fails` from `ERROR` to `INFO`) #12383

- DDL
  - Add the `tidb_allow_remove_auto_inc` variable. Dropping the `AUTO INCREMENT` attribute of the column is disabled by default #12145
  - Fix the issue that the uncommented TiDB-specific syntax `PRE_SPLIT_REGIONS` might cause errors in the downstream database during data replication #12120
  - Add the `split-region-max-num` variable in the configuration file so that the maximum allowable number of Regions is adjustable #12097
  - Support splitting a Region into multiple Regions and fix the timeout issue during Region scatterings #12343
  - Fix the issue that the `drop index` statement fails when the index that contains an `AUTO_INCREMENT` column referenced by two indexes #12344
- Monitor
  - Add the `connection_transient_failure_count` monitoring metrics to count the number of gRPC connection errors in `tikvclient` #12093

9.2.17.2 TiKV

- Raftstore
  - Fix the issue that Raftstore inaccurately counts the number of keys in empty Regions #5414
  - Support double linked list for RocksDB to improve the performance of reverse scan #5368
  - Support batch Region split command and empty split command to improve split performance #5470
- Server
  - Fix the issue that the output format of the `-V` command is not consistent with the format of 2.X #5501
  - Upgrade Titan to the latest version in the 3.0 branch #5517
  - Upgrade grpcio to v0.4.5 #5523
  - Fix the issue of gRPC coredump and support shared memory to avoid OOM #5524
  - Fix the issue in TiKV that file descriptor leak in idle clusters might cause TiKV processes to exit abnormally when the processes run for a long time #5567
- Storage
  - Support the `txn_heart_beat` API to make the pessimistic lock in TiDB consistent with that in MySQL as much as possible #5507
  - Fix the issue that the performance of point queries is low in some situations #5495 #5463

9.2.17.3 PD

- Fix the issue that adjacent small Regions cannot be merged #1726
- Fix the issue that the TLS enabling parameter in `pd-ctl` is invalid #1738
- Fix the thread-safety issue that the PD operator is accidentally removed #1734
- Support TLS for Region syncer #1739

9.2.17.4 Tools

- TiDB Binlog
  - Add the `worker-count` and `txn-batch` configuration items in Reparo to control the recovery speed #746
  - Optimize the memory usage of Drainer to enhance the efficiency of simultaneous execution #737
- TiDB Lightning
  - Fix the issue that re-importing data from checkpoint might cause TiDB Lightning to panic #237
  - Optimize the algorithm of `AUTO_INCREMENT` to reduce the risk of overflowing `AUTO_INCREMENT` columns #227

9.2.17.5 TiDB Ansible

- Upgrade TiSpark to v2.2.0 #926
- Update the default value of the TiDB configuration item `pessimistic_txn` to `true` #933
- Add more system-level monitoring metrics to `node_exporter` #938
- Add two perf tools `iosnoop` and `funcslower` in TiDB Ansible to better diagnose the cluster state #946
- Replace the raw module to shell module to address the long waiting time in such situations as the password expires #949
- Update the default value of the TiDB configuration item `txn_local_latches` to `false`
- Optimize the monitoring metrics and alert rules of Grafana dashboard #962 #963 #969
- Check the configuration file before the deployment and upgrade #934 #972

9.2.18 TiDB 3.0.3 Release Notes

Release date: August 29, 2019

TiDB version: 3.0.3

TiDB Ansible version: 3.0.3

9.2.18.1 TiDB

- SQL Optimizer
  - Add the `opt_rule_blacklist` table to disable logic optimization rules such as `aggregation_eliminate` and `column_prune` #11658
  - Fix the issue that incorrect results might be returned for `Index Join` when the join key uses a prefix index or an unsigned index column that is equal to a negative value #11759
  - Fix the issue that `"` or `\` in the `SELECT` statements of `create ⋯ binding ...` might result in parsing errors #11726
- SQL Execution Engine
  - Fix the issue that type errors in the return value might occur when the Quote function handles a null value #11619
  - Fix the issue that incorrect results for `ifnull` might be returned when Max/Min is used for type inferring with `NotNullFlag` retained #11641
  - Fix the potential error that occurs when comparing bit type data in string form #11660
  - Decrease the concurrency for data that requires sequential read to lower the possibility of OOM #11679
  - Fix the issue that incorrect type inferring might be caused when multiple parameters are unsigned for some built-in functions (for example, `if` and `coalesce`) #11621
  - Fix the incompatibility with MySQL when the `Div` function handles unsigned decimal types #11813
  - Fix the issue that panic might occur when executing SQL statements that modify the status of Pump/Drainer #11827
  - Fix the issue that panic might occur for `select ... for update` when Autocommit = 1 and there is no `begin` statement #11736
  - Fix the permission check error that might occur when the `set default role` statement is executed #11777
  - Fix the permission check error that might occur when `create user` or `drop user` is executed #11814
  - Fix the issue that the `select ... for update` statement might auto retry when it is constructed into the `PointGetExecutor` function #11718
  - Fix the boundary error that might occur when the Window function handles partition #11825
  - Fix the issue that the `time` function hits EOF errors when handling an incorrectly formatted argument #11893
  - Fix the issue that the Window function does not check the passed-in parameters #11705
  - Fix the issue that the plan result viewed via `Explain` is inconsistent with the actually executed plan #11186

  - Fix the issue that duplicate memory referenced by the Window function might result in a crash or incorrect results #11823
  - Update the incorrect information in the `Succ` field in the slow log #11887

- Server
  - Rename the `tidb_back_off_wexight` variable to `tidb_backoff_weight` #11665
  - Update the minimum TiKV version compatible with the current TiDB to v3.0.0 #11618
  - Support `make testSuite` to ensure the suites in the test are correctly used #11685

- DDL
  - Skip the execution of unsupported partition-related DDL statements, including statements that modify the partition type while deleting multiple partitions #11373
  - Disallow a Generated Column to be placed before its dependent columns #11686
  - Modify the default values of `tidb_ddl_reorg_worker_cnt` and `tidb_ddl_reorg_batch_size` ↪ #11874

- Monitor
  - Add new backoff monitoring types to record duration for each backoff type; add more backoff metrics to cover previously uncounted types such as commit backoff #11728

9.2.18.2 TiKV

- Fix the issue that ReadIndex might fail to respond to requests because of duplicate context #5256
- Fix potential scheduling jitters caused by premature `PutStore` #5277
- Fix incorrect timestamps reported from Region heartbeats #5296
- Reduce the size of core dump by excluding the shared block cache from it #5322
- Fix potential TiKV panics during region merge #5291
- Speed up leader change check for the dead lock detector #5317
- Support using `grpc env` to create deadlock clients #5346
- Add `config-check` to check whether the configuration is correct #5349
- Fix the issue that ReadIndex does not return anything when there is no leader #5351

9.2.18.3 PD

- Return success message for `pdctl` #1685

9.2.18.4 Tools

- TiDB Binlog
  - Modify the default value of `defaultBinlogItemCount` in Drainer from 65536 to 512 to reduce the chance of OOM on Drainer startup #721
  - Optimize the offline logic for pump server to avoid potential offline congestions #701
- TiDB Lightning:
  - Skip the system databases `mysql`, `information_schema`, `performance_schema`, and `sys` by default when importing #225

9.2.18.5 TiDB Ansible

- Optimize PD operations for rolling update to improve stability #894
- Remove the Grafana Collector components that are not supported by the current Grafana version #892
- Update TiKV alerting rules #898
- Fix the issue that the generated TiKV configuration misses the `pessimistic-txn` parameter #911
- Update Spark to V2.4.3, and update TiSpark to V2.1.4 that is compatible with Spark V2.4.3 #913, #918

9.2.19 TiDB 3.0.2 Release Notes

Release date: August 7, 2019

TiDB version: 3.0.2

TiDB Ansible version: 3.0.2

9.2.19.1 TiDB

- SQL Optimizer
  - Fix the issue that the "Can't find column in schema" message is reported when the same table occurs multiple times in a query and logically the query result is always empty #11247
  - Fix the issue that the query plan does not meet the expectation caused by the `TIDB_INLJ` hint not working correctly in some cases (like `explain select /*+ TIDB_INLJ(t1)*/ t1.b, t2.a from t t1, t t2 where t1.b = t2.a`) #11362
  - Fix the issue that the column name in the query result is wrong in some cases (like `SELECT IF(1,c,c)FROM t`) #11379

  - Fix the issue that some queries like `SELECT 0 LIKE 'a string'` return `TRUE` because the `LIKE` expression is implicitly converted to 0 in some cases #11411
  - Support sub-queries in the `SHOW` statement, like `SHOW COLUMNS FROM tbl WHERE FIELDS IN (SELECT 'a')` #11459
  - Fix the issue that the related column of the aggregate function cannot be found and an error is reported caused by the `outerJoinElimination` optimizing rule not correctly handling the column alias; improve alias parsing in the optimizing process to make optimization cover more query types #11377
  - Fix the issue that no error is reported when the syntax restriction is violated in the Window function (for example, `UNBOUNDED PRECEDING` is not allowed to appear at the end of the Frame definition) #11543
  - Fix the issue that `FUNCTION_NAME` is in uppercase in the `ERROR 3593 (HY000): You cannot use the window function FUNCTION_NAME in this context` error message, which causes incompatibility with MySQL #11535
  - Fix the issue that the unimplemented `IGNORE NULLS` syntax in the Window function is used but no error is reported #11593
  - Fix the issue that the Optimizer does not correctly estimate time equal conditions #11512
  - Support updating the Top-N statistics based on the feedback information #11507

- SQL Execution Engine

  - Fix the issue that the returned value is not `NULL` when the `INSERT` function contains `NULL` in parameters #11248
  - Fix the issue that the computing result might be wrong when the partitioned table is checked by the `ADMIN CHECKSUM` operation #11266
  - Fix the issue that the result might be wrong when INDEX JOIN uses the prefix index #11246
  - Fix the issue that result might be wrong caused by incorrectly aligning fractions when the `DATE_ADD` function does subtraction on date numbers involving microseconds #11288
  - Fix the wrong result caused by the `DATE_ADD` function incorrectly processing the negative numbers in `INTERVAL` #11325
  - Fix the issue that the number of fractional digits returned by `Mod(%)`, `Multiple (*)` or `Minus(-)` is different from that in MySQL when `Mod(%)`, `Multiple(*)` or `Minus(-)` returns 0 and the number of fractional digits is large (like `select 0.000 % 0.11234500000000000000`) #11251
  - Fix the issue that `NULL` with a warning is incorrectly returned when the length of the result returned by `CONCAT` and `CONCAT_WS` functions exceeds `max_allowed_packet` #11275
  - Fix the issue that `NULL` with a warning is incorrectly returned when parameters in the `SUBTIME` and `ADDTIME` functions are invalid #11337
  - Fix the issue that `NULL` is incorrectly returned when parameters in the `CONVERT_TZ` function are invalid #11359
  - Add the `MEMORY` column to the result returned by `EXPLAIN ANALYZE` to show the memory usage of this query #11418

  - Add `CARTESIAN` Join to the result of `EXPLAIN` #11429
  - Fix the incorrect data of auto-increment columns of the float and double types #11385
  - Fix the panic issue caused by some `nil` information when pseudo statistics are dumped #11460
  - Fix the incorrect query result of `SELECT ⋯ CASE WHEN ⋯ ELSE NULL ...` caused by constant folding optimization #11441
  - Fix the issue that `floatStrToIntStr` does not correctly parse the input such as `+999.9999e2` #11473
  - Fix the issue that `NULL` is not returned in some cases when the result of the `DATE_ADD` and `DATE_SUB` function overflows #11476
  - Fix the issue that the conversion result is different from that in MySQL if the string contains an invalid character when a long string is converted to an integer #11469
  - Fix the issue that the result of the `REGEXP BINARY` function is incompatible with MySQL caused by case sensitiveness of this function #11504
  - Fix the issue that an error is reported when the `GRANT ROLE` statement receives `CURRENT_ROLE`; fix the issue that the `REVOKE ROLE` statement does not correctly revoke the `mysql.default_role` privilege #11356
  - Fix the display format issue of the `Incorrect datetime value` warning information when executing statements like `SELECT ADDDATE('2008-01-34', -1)` #11447
  - Fix the issue that the error message reports `constant ⋯ overflows float` ↪ rather than `constant ⋯ overflows bigint` if the result overflows when a float field of the JSON data is converted to an integer #11534
  - Fix the issue that the result might be wrong caused by incorrect type conversion when the `DATE_ADD` function receives `FLOAT`, `DOUBLE` and `DECIMAL` column parameters #11527
  - Fix the wrong result caused by incorrectly processing the sign of the INTERVAL fraction in the `DATE_ADD` function #11615
  - Fix the incorrect query result when Index Lookup Join contains the prefix index caused by `Ranger` not correctly handling the prefix index #11565
  - Fix the issue that the "Incorrect arguments to NAME_CONST" message is reported if the `NAME_CONST` function is executed when the second parameter of `NAME_CONST` is a negative number #11268
  - Fix the issue that the result is incompatible with MySQL when an SQL statement involves computing the current time and the value is fetched multiple times; use the same value when fetching the current time for the same SQL statement #11394
  - Fix the issue that `Close` is not called for `ChildExecutor` when the `Close` of `baseExecutor` reports an error. This issue might lead to Goroutine leaks when the `KILL` statements do not take effect and `ChildExecutor` is not closed #11576

- Server
  - Fix the issue that the auto-added value is 0 instead of the current timestamp when `LOAD DATA` processes the missing `TIMESTAMP` field in the CSV file #11250

  - Fix issues that the `SHOW CREATE USER` statement does not correctly check related privileges, and `USER` and `HOST` returned by `SHOW CREATE USER CURRENT_USER()` might be wrong #11229
  - Fix the issue that the returned result might be wrong when `executeBatch` is used in JDBC #11290
  - Reduce printing the log information of the streaming client when changing the TiKV server's port #11370
  - Optimize the logic of reconnecting the streaming client to the TiKV server so that the streaming client will not be blocked for a long time #11372
  - Add `REGION_ID` in `INFORMATION_SCHEMA.TIDB_HOT_REGIONS` #11350
  - Cancel the timeout duration of obtaining Region information from the PD API to ensure that obtaining Region information will not end in a failure when TiDB API `http://{TiDBIP}:10080/regions/hot` is called due to PD timeout when the number of Regions is large #11383
  - Fix the issue that Region related requests do not return partitioned table-related Regions in the HTTP API #11466
  - Make the following changes to reduce the probability of locking timeout caused by slow operations when the user manually validates pessimistic locking #11521:
    * Increase the default TTL of pessimistic locking from 30 seconds to 40 seconds
    * Increase the maximum TTL from 60 seconds to 120 seconds
    * Calculate the pessimistic locking duration from the first `LockKeys` request
  - Change the `SendRequest` function logic in the TiKV client: try to immediately connect to another peer instead of keeping waiting when the connect cannot be built #11531
  - Optimize the Region cache: label the removed store as invalid when a store is moved while another store goes online with a same address, to update the store information in the cache as soon as possible #11567
  - Add the Region ID to the result returned by the `http://{TiDB_ADDRESS:TIDB_IP }/mvcc/key/{db}/{table}/{handle}` API #11557
  - Fix the issue that Scatter Table does not work caused by the Scatter Table API not escaping the Range key #11298
  - Optimize the Region cache: label the store where the Region exists as invalid when the correspondent store is inaccessible, to avoid reduced query performance caused by accessing this store #11498
  - Fix the error that the table schema can still be obtained through the HTTP API after dropping the database with the same name multiple times #11585

- DDL
  - Fix the issue that an error occurs when a non-string column with a zero length is being indexed #11214
  - Disallow modifying the columns with foreign key constraints and full-text indexes (Note: TiDB still supports foreign key constraints and full-text indexes in syntax) #11274
  - Fix the issue that the index offset of the column might be wrong because the position changed by the `ALTER TABLE` statement and the default value of the

column are used concurrently #11346
  - Fix two issues that occur when parsing JSON files:
    * `int64` is used as the intermediate parsing result of `uint64` in `ConvertJSONToFloat` ↪ , which leads to the precision overflow error #11433
    * `int64` is used as the intermediate parsing result of `uint64` in `ConvertJSONToInt` ↪ , which leads to the precision overflow error #11551
  - Disallow dropping indexes on the auto-increment column to avoid that the auto-increment column might get an incorrect result #11399
  - Fix the following issues #11492:
    * The character set and the collation of the column are not consistent when explicitly specifying the collation but not the character set
    * The error is not correctly reported when there is a conflict between the character set and the collation that are specified by `ALTER TABLE ⋯ MODIFY` ↪ `COLUMN`
    * Incompatibility with MySQL when using `ALTER TABLE ⋯ MODIFY COLUMN` to specify character sets and collations multiple times
  - Add the trace details of the subquery to the result of the `TRACE` query #11458
  - Optimize the performance of executing `ADMIN CHECK TABLE` and greatly reduce its execution time #11547
  - Add the result returned by `SPLIT TABLE ⋯ REGIONS/INDEX` and make `TOTAL_SPLIT_REGION` and `SCATTER_FINISH_RATIO` display the number of Regions that have been split successfully before timeout in the result #11484
  - Fix the issue that the precision displayed by statements like `SHOW CREATE TABLE` is incomplete when `ON UPDATE CURRENT_TIMESTAMP` is the column attribute and the float precision is specified #11591
  - Fix the issue that the index result of the column cannot be correctly calculated when the expression of a virtual generated column contains another virtual generated column #11475
  - Fix the issue that the minus sign cannot be added after `VALUE LESS THAN` in the `ALTER TABLE ⋯ ADD PARTITION ⋯` statement #11581
- Monitor
  - Fix the issue that data is not collected and reported because the `TiKVTxnCmdCounter` ↪ monitoring metric is not registered #11316
  - Add the `BindUsageCounter`, `BindTotalGauge` and `BindMemoryUsage` monitoring metrics for the Bind Info #11467

9.2.19.2 TiKV

- Fix the bug that TiKV panics if the Raft log is not written in time #5160
- Fix the bug that the panic information is not written into the log file after TiKV panics #5198
- Fix the bug that the Insert operation might be incorrectly performed in the pessimistic transaction #5203

- Lower the output level of some logs that require no manual intervention to INFO #5193
- Improve the accuracy of monitoring the storage engine size #5200
- Improve the accuracy of the Region size in tikv-ctl #5195
- Improve the performance of the deadlock detector for pessimistic locking #5192
- Improve the performance of GC in the Titan storage engine #5197

9.2.19.3 PD

- Fix the bug that the Scatter Region scheduler cannot work #1642
- Fix the bug that the merge Region operation cannot be performed in pd-ctl #1653
- Fix the bug that the remove-tombstone operation cannot be performed in pd-ctl #1651
- Fix the issue that the Region overlapping with the key scope cannot be found when performing the scan Region operation #1648
- Add the retrying mechanism to make sure that the members are added successfully in PD #1643

9.2.19.4 Tools

TiDB Binlog

- Add the configuration item check feature when starting, which will stop the Binlog service and report an error when an invalid item is found #687
- Add the `node-id` configuration in Drainer to specify a specific logic used by Drainer #684

TiDB Lightning

- Fix the issue that `tikv_gc_life_time` fails to be changed back to its original value when 2 checksums are running at the same time #218
- Add the configuration item check feature when starting, which will stop the Binlog service and report an error when an invalid item is found #217

9.2.19.5 TiDB Ansible

- Fix the unit error that the Disk Performance monitor treats seconds as milliseconds #840
- Add the `log4j` configuration file in Spark #841
- Fix the issue that the Prometheus configuration file is generated in the wrong format when Binlog is enabled and Kafka or ZooKeeper is configured #844
- Fix the issue that the `pessimistic-txn` configuration parameter is left out in the generated TiDB configuration file #850
- Add and optimize metrics on the TiDB Dashboard #853
- Add descriptions for each monitoring item on the TiDB Dashboard #854

- Add the TiDB Summary Dashboard to better view the cluster status and troubleshoot issues #855
- Update the Allocator Stats monitoring item on the TiKV Dashboard #857
- Fix the unit error in the Node Exporter's alerting expression #860
- Upgrade the TiSpark jar package to v2.1.2 #862
- Update the descriptions of the Ansible Task feature #867
- Update the expression of the local reader requests monitoring item on the TiDB Dashboard #874
- Update the expression of the TiKV Memory monitoring item on the Overview Dashboard, and fix the issue of wrongly displayed monitoring #879
- Remove the Binlog support in the Kafka mode #878
- Fix the issue that PD fails to transfer the Leader when executing the `rolling_update ↪ .yml` operation #887

9.2.20 TiDB 3.0.1 Release Notes

Release date: July 16, 2019

TiDB version: 3.0.1

TiDB Ansible version: 3.0.1

9.2.20.1 TiDB

- Add support for the `MAX_EXECUTION_TIME` feature #11026
- Add the `tidb_wait_split_region_finish_backoff` session variable to control the backoff time of splitting Regions #11166
- Support automatically adjusting the incremental gap allocated by auto-increment IDs based on the load, and the auto-adjustment scope of the incremental gap is 1000~2000000 #11006
- Add the `ADMIN PLUGINS ENABLE/ADMIN PLUGINS DISABLE` SQL statement to dynamically enable or disable plugins #11157
- Add the session connection information in the Audit plugin #11013
- Change the default behavior during the period of splitting Regions to wait for PD to finish scheduling #11166
- Prohibit Window Functions from being cached in Prepare Plan Cache to avoid incorrect results in some cases #11048
- Prohibit `ALTER` statements from modifying the definition of stored generated columns #11068
- Disallow changing virtual generated columns to stored generated columns #11068
- Disallow changing the generated column expression with indexes #11068
- Support compiling TiDB on the ARM64 architecture #11150
- Support modifying the collation of a database or a table, but the character set of the database/table has to be UTF-8 or utf8mb4 #11086

- Fix the issue that an error is reported when the `SELECT` subquery in the `UPDATE ⋯ SELECT` statement fails to parse the column in the `UPDATE` expression and the column is wrongly pruned #11252
- Fix the panic issue that happens when a column is queried on multiple times and the returned result is NULL during point queries #11226
- Fix the data race issue caused by non-thread safe `rand.Rand` when using the `RAND` function #11169
- Fix the bug that the memory usage of a SQL statement exceeds the threshold but the execution of this statement is not canceled in some cases when `oom-action="cancel"` is configured, and the returned result is incorrect #11004
- Fix the issue that `SHOW PROCESSLIST` shows that the memory usage is not `0` because the memory usage of MemTracker was not correctly cleaned #10970
- Fix the bug that the result of comparing integers and non-integers is not correct in some cases #11194
- Fix the bug that the query result is not correct when the query on table partitions contains a predicate in explicit transactions #11196
- Fix the DDL job panic issue because `infoHandle` might be `NULL` #11022
- Fix the issue that the query result is not correct because the queried column is not referenced in the subquery and is then wrongly pruned when running a nested aggregation query #11020
- Fix the issue that the `Sleep` function does not respond to the `KILL` statement in time #11028
- Fix the issue that the `DB` and `INFO` columns shown by the `SHOW PROCESSLIST` command are incompatible with MySQL #11003
- Fix the system panic issue caused by the `FLUSH PRIVILEGES` statement when `skip-grant-table=true` is configured #11027
- Fix the issue that the primary key statistics collected by `FAST ANALYZE` are not correct when the table primary key is an `UNSIGNED` integer #11099
- Fix the issue that the "invalid key" error is reported by the `FAST ANALYZE` statement in some cases #11098
- Fix the issue that the precision shown by the `SHOW CREATE TABLE` statement is incomplete when `CURRENT_TIMESTAMP` is used as the default value of the column and the float precision is specified #11088
- Fix the issue that the function name is not in lowercase when window functions report an error to make it compatible with MySQL #11118
- Fix the issue that TiDB fails to connect to TiKV and thus cannot provide service after the background thread of TiKV Client Batch gRPC panics #11101
- Fix the issue that the variable is set incorrectly by `SetVar` because of the shallow copy of the string #11044
- Fix the issue that the execution fails and an error is reported when the `INSERT ⋯ ON DUPLICATE` statement is applied on table partitions #11231
- Pessimistic locking (experimental feature)
    - Fix the issue that an incorrect result is returned because of the invalid lock on the row when point queries are run using the pessimistic locking and the returned

data is empty #10976
  - Fix the issue that the query result is not correct because `SELECT ⋯ FOR UPDATE` does not use the correct TSO when using the pessimistic locking in the query #11015
  - Change the detection behavior from immediate conflict detection to waiting when an optimistic transaction meets a pessimistic lock to avoid worsening the lock conflict #11051

9.2.20.2 TiKV

- Add the statistics of the size of blob files in statistics information #5060
- Fix the core dump issue caused by the incorrectly cleaned memory resources when the process exits #5053
- Add all monitoring metrics related to the Titan engine #4772, #4836
- Add the number of open file handles for Titan when counting the number of open file handles to avoid the issue that no file handle is available because of inaccurate statistics of file handles #5026
- Set `blob_run_mode` to decide whether to enable the Titan engine on a specific CF #4991
- Fix the issue that the read operations cannot get the commit information of pessimistic transactions #5067
- Add the `blob-run-mode` configuration parameter to control the running mode of the Titan engine, and its value can be `normal`, `read-only` or `fallback` #4865
- Improve the performance of detecting deadlocks #5089

9.2.20.3 PD

- Fix the issue that the scheduling limit is automatically adjusted to 0 when PD schedules hot Regions #1552
- Add the `enable-grpc-gateway` configuration option to enable the gRPC gateway feature of etcd #1596
- Add `store-balance-rate`, `hot-region-schedule-limit` and other statistics related to scheduler configuration #1601
- Optimize the hot Region scheduling strategy and skip the Regions that lack replicas during scheduling to prevent multiple replicas from being scheduled to the same IDC #1609
- Optimize the Region merge processing logic and support giving priority to merging the Regions with smaller sizes to speed up Region merging #1613
- Adjust the default limit of hot Region scheduling in a single time to 64 to prevent too many scheduling tasks from occupying system resources and impacting performance #1616
- Optimize the Region scheduling strategy and support giving high priority to scheduling Regions in the `Pending` status #1617

- Fix the issue that `random-merge` and `admin-merge-region` operators cannot be added #1634
- Adjust the format of the Region key in the log to hexadecimal notation to make it easier to view #1639

9.2.20.4 Tools

TiDB Binlog

- Optimize the Pump GC strategy and remove the restriction that the unconsumed binlog cannot be cleaned to make sure that the resources are not occupied for a long time #646

TiDB Lightning

- Fix the import error that happens when the column names specified by the SQL dump are not in lowercase #210

9.2.20.5 TiDB Ansible

- Add the precheck feature for the ansible command and its `jmespath` and `jinja2` dependency packages #803, #813
- Add the `stop-write-at-available-space` parameter (10 GiB by default) in Pump to stop writing binlog files in Pump when the available disk space is less than the parameter value #806
- Update the I/O monitoring items in the TiKV monitoring information and make them compatible with the monitoring components of the new version #820
- Update the PD monitoring information, and fix the anomaly that Disk Latency is empty in the disk performance dashboard #817
- Add monitoring items for Titan in the TiKV details dashboard #824

9.2.21 TiDB 3.0 GA Release Notes

Release date: June 28, 2019

TiDB version: 3.0.0

TiDB Ansible version: 3.0.0

9.2.21.1 Overview

On June 28, 2019, TiDB 3.0 GA is released. The corresponding TiDB Ansible version is 3.0.0. Compared with TiDB 2.1, this release has greatly improved in the following aspects:

- Stability. TiDB 3.0 has demonstrated long-term stability for large-scale clusters with up to 150+ nodes and 300+ TB of storage.
- Usability. TiDB 3.0 has multi-facet improvements in usability, including standardized slow query logs, well-developed log file specification, and new features such as `EXPLAIN ANALYZE` and SQL Trace to save operation costs for users.
- Performance. The performance of TiDB 3.0 is 4.5 times greater than TiDB 2.1 in TPC-C benchmarks, and over 1.5 times in Sysbench benchmarks. Thanks to the support for Views, TPC-H 50G Q15 can now run normally.
- New features including Window Functions, Views (**Experimental**), partitioned tables, the plugin framework, pessimistic locking (**Experimental**), and `SQL Plan Management`.

9.2.21.2 TiDB

- New Features
  - Support Window Functions; compatible with all window functions in MySQL 8.0, including `NTILE`, `LEAD`, `LAG`, `PERCENT_RANK`, `NTH_VALUE`, `CUME_DIST`, `FIRST_VALUE`, `LAST_VALUE`, `RANK`, `DENSE_RANK`, and `ROW_NUMBER`
  - Support Views (**Experimental**)
  - Improve Table Partition
    - Support Range Partition
    - Support Hash Partition
  - Add the plug-in framework, supporting plugins such as IP Whitelist (**Enterprise**) and Audit Log (**Enterprise**).
  - Support the SQL Plan Management function to create SQL execution plan binding to ensure query stability (**Experimental**)
- SQL Optimizer
  - Optimize the `NOT EXISTS` subquery and convert it to `Anti Semi Join` to improve performance
  - Optimize the constant propagation on the `Outer Join`, and add the optimization rule of `Outer Join` elimination to reduce non-effective computations and improve performance
  - Optimize the `IN` subquery to execute `Inner Join` after aggregation to improve performance
  - Optimize `Index Join` to adapt to more scenarios
  - Improve the Partition Pruning optimization rule of Range Partition
  - Optimize the query logic for `_tidb_rowid`to avoid full table scan and improve performance
  - Match more prefix columns of the indexes when extracting access conditions of composite indexes if there are relevant columns in the filter to improve performance
  - Improve the accuracy of cost estimates by using order correlation between columns

  - Optimize `Join Order` based on the greedy strategy and the dynamic programming algorithm to speed up the join operation of multiple tables
  - Support Skyline Pruning, with some rules to prevent the execution plan from relying too heavily on statistics to improve query stability
  - Improve the accuracy of row count estimation for single-column indexes with NULL values
  - Support `FAST ANALYZE` that randomly samples in each Region to avoid full table scan and improve performance with statistics collection
  - Support the incremental Analyze operation on monotonically increasing index columns to improve performance with statistics collection
  - Support using subqueries in the `DO` statement
  - Support using `Index Join` in transactions
  - Optimize `prepare`/`execute` to support DDL statements with no parameters
  - Modify the system behavior to auto load statistics when the `stats-lease` variable value is 0
  - Support exporting historical statistics
  - Support the `dump`/`load` correlation of histograms

- SQL Execution Engine

  - Optimize log output: `EXECUTE` outputs user variables and `COMMIT` outputs slow query logs to facilitate troubleshooting
  - Support the `EXPLAIN ANALYZE` function to improve SQL tuning usability
  - Support the `admin show next_row_id` command to get the ID of the next row
  - Add six built-in functions: `JSON_QUOTE`, `JSON_ARRAY_APPEND`, `JSON_MERGE_PRESERVE` ↪ , `BENCHMARK` ,`COALESCE`, and `NAME_CONST`
  - Optimize control logics on the chunk size to dynamically adjust based on the query context, to reduce the SQL execution time and resource consumption
  - Support tracking and controlling memory usage in three operators - `TableReader`, `IndexReader` and `IndexLookupReader`
  - Optimize the Merge Join operator to support an empty `ON` condition
  - Optimize write performance for single tables that contains too many columns
  - Improve the performance of `admin show ddl jobs` by supporting scanning data in reverse order
  - Add the `split table region` statement to manually split the table Region to alleviate hotspot issues
  - Add the `split index region` statement to manually split the index Region to alleviate hotspot issues
  - Add a blocklist to prohibit pushing down expressions to Coprocessor
  - Optimize the `Expensive Query` log to print the SQL query in the log when it exceeds the configured limit of execution time or memory

- DDL

  - Support migrating from character set `utf8` to `utf8mb4`
  - Change the default character set from`utf8` to `utf8mb4`

  - Add the `alter schema` statement to modify the character set and the collation of the database
  - Support ALTER algorithm `INPLACE/INSTANT`
  - Support `SHOW CREATE VIEW`
  - Support `SHOW CREATE USER`
  - Support fast recovery of mistakenly deleted tables
  - Support adjusting the number of concurrencies of ADD INDEX dynamically
  - Add the `pre_split_regions` option that pre-allocates Regions when creating the table using the `CREATE TABLE` statement, to relieve write hot Regions caused by lots of writes after the table creation
  - Support splitting Regions by the index and range of the table specified using SQL statements to relieve hotspot issues
  - Add the `ddl_error_count_limit` global variable to limit the number of DDL task retries
  - Add a feature to use `SHARD_ROW_ID_BITS` to scatter row IDs when the column contains an AUTO_INCREMENT attribute to relieve hotspot issues
  - Optimize the lifetime of invalid DDL metadata to speed up recovering the normal execution of DDL operations after upgrading the TiDB cluster

- Transactions
  - Support the pessimistic transaction model (**Experimental**)
  - Optimize transaction processing logics to adapt to more scenarios:
    * Change the default value `tidb_disable_txn_auto_retry` to `on`, which means non-auto committed transactions will not be retried
    * Add the `tidb_batch_commit` system variable to split a transaction into multiple ones to be executed concurrently
    * Add the `tidb_low_resolution_tso` system variable to control the number of TSOs to obtain in batches and reduce the number of times that transactions request for TSOs, to improve performance in scenarios with relatively low requirement of consistency
    * Add the `tidb_skip_isolation_level_check` variable to control whether to report errors when the isolation level is set to SERIALIZABLE
    * Modify the `tidb_disable_txn_auto_retry` system variable to make it work on all retryable errors

- Permission Management
  - Perform permission check on the `ANALYZE`, `USE`, `SET GLOBAL`, and `SHOW PROCESSLIST` statements
  - Support Role Based Access Control (RBAC) (**Experimental**)

- Server
  - Optimize slow query logs:
    * Restructure the log format
    * Optimize the log content

    * Optimize the log query method to support using the `INFORMATION_SCHEMA .SLOW_QUERY` and `ADMIN SHOW SLOW` statements of the memory table to query slow query logs
  - Develop a unified log format specification with restructured log system to facilitate collection and analysis by tools
  - Support using SQL statements to manage TiDB Binlog services, including querying status, enabling TiDB Binlog, maintaining and sending TiDB Binlog strategies.
  - Support using `unix_socket` to connect to the database
  - Support `Trace` for SQL statements
  - Support getting information for a TiDB instance via the `/debug/zip` HTTP interface to facilitate troubleshooting.
  - Optimize monitoring items to facilitate troubleshooting:
    * Add the `high_error_rate_feedback_total` monitoring item to monitor the difference between the actual data volume and the estimated data volume based on statistics
    * Add a QPS monitoring item in the database dimension
  - Optimize the system initialization process to only allow the DDL owner to perform the initialization. This reduces the startup time for initialization or upgrading.
  - Optimize the execution logic of `kill query` to improve performance and ensure resource is release properly
  - Add a startup option `config-check` to check the validity of the configuration file
  - Add the `tidb_back_off_weight` system variable to control the backoff time of internal error retries
  - Add the `wait_timeout`and `interactive_timeout` system variables to control the maximum idle connections allowed
  - Add the connection pool for TiKV to shorten the connection establishing time

- Compatibility

  - Support the `ALLOW_INVALID_DATES` SQL mode
  - Support the MySQL 320 Handshake protocol
  - Support manifesting unsigned BIGINT columns as auto-increment columns
  - Support the `SHOW CREATE DATABASE IF NOT EXISTS` syntax
  - Optimize the fault tolerance of `load data` for CSV files
  - Abandon the predicate pushdown operation when the filtering condition contains a user variable to improve the compatibility with MySQL's behavior of using user variables to simulate Window Functions

9.2.21.3 PD

- Support re-creating a cluster from a single node
- Migrate Region metadata from etcd to the go-leveldb storage engine to solve the storage bottleneck in etcd for large-scale clusters

- API
  - Add the `remove-tombstone` API to clear Tombstone stores
  - Add the `ScanRegions` API to batch query Region information
  - Add the `GetOperator` API to query running operators
  - Optimize the performance of the `GetStores` API
- Configurations
  - Optimize configuration check logic to avoid configuration item errors
  - Add `enable-two-way-merge` to control the direction of Region merge
  - Add `hot-region-schedule-limit` to control the scheduling rate for hot Regions
  - Add `hot-region-cache-hits-threshold` to identify hotspot when hitting multiple thresholds consecutively
  - Add the `store-balance-rate` configuration item to control the maximum numbers of balance Region operators allowed per minute
- Scheduler Optimizations
  - Add the store limit mechanism for separately controlling the speed of operators for each store
  - Support the `waitingOperator` queue to optimize the resource race among different schedulers
  - Support scheduling rate limit to actively send scheduling operations to TiKV. This improves the scheduling rate by limiting the number of concurrent scheduling tasks on a single node.
  - Optimize the `Region Scatter` scheduling to be not restrained by the limit mechanism
  - Add the `shuffle-hot-region` scheduler to facilitate TiKV stability test in scenarios of poor hotspot scheduling
- Simulator
  - Add simulator for data import scenarios
  - Support setting different heartbeats intervals for the Store
- Others
  - Upgrade etcd to solve the issues of inconsistent log output formats, Leader selection failure in prevote, and lease deadlocking
  - Develop a unified log format specification with restructured log system to facilitate collection and analysis by tools
  - Add monitoring metrics including scheduling parameters, cluster label information, time consumed by PD to process TSO requests, Store ID and address information, etc.

9.2.21.4 TiKV

- Support distributed GC and concurrent lock resolving for improved GC performance

- Support reversed `raw_scan` and `raw_batch_scan`
- Support Multi-thread Raftstore and Multi-thread Apply to improve scalabilities, concurrency capacity, and resource usage within a single node. Performance improves by 70% under the same level of pressure
- Support batch receiving and sending Raft messages, improving TPS by 7% for write intensive scenarios
- Support checking RocksDB Level 0 files before applying snapshots to avoid write stall
- Introduce Titan, a key-value plugin that improves write performance for scenarios with value sizes greater than 1KiB, and relieves write amplification in certain degrees
- Support the pessimistic transaction model (**Experimental**)
- Support getting monitoring information via HTTP
- Modify the semantics of `Insert` to allow Prewrite to succeed only when there is no Key
- Develop a unified log format specification with restructured log system to facilitate collection and analysis by tools
- Add performance metrics related to configuration information and key bound crossing
- Support Local Reader in RawKV to improve performance
- Engine
  - Optimize memory management to reduce memory allocation and copying for `Iterator Key Bound Option`
  - Support `block cache` sharing among different column families
- Server
  - Reduce context switch overhead from `batch commands`
  - Remove `txn scheduler`
  - Add monitoring items related to `read index` and `GC worker`
- RaftStore
  - Support Hibernate Regions to optimize CPU consumption from RaftStore (**Experimental**)
  - Remove the local reader thread
- Coprocessor
  - Refactor the computation framework to implement vector operators, computation using vector expressions, and vector aggregations to improve performance
  - Support providing operator execution status for the `EXPLAIN ANALYZE` statement in TiDB
  - Switch to the `work-stealing` thread pool model to reduce context switch cost

9.2.21.5 Tools

- TiDB Lightning
  - Support redirected replication of data tables

  - Support importing CSV files
  - Improve performance for conversion from SQL to KV pairs
  - Support batch import of single tables to improve performance
  - Support separately importing data and indexes for big tables to improve the performance of TiKV-importer
  - Support filling the missing column using the `row_id` or the default column value when column data is missing in the new file
  - Support setting a speed limit in `TIKV-importer` when uploading SST files to TiKV

- TiDB Binlog

  - Add the `advertise-addr` configuration in Drainer to support the bridge mode in the container environment
  - Add the `GetMvccByEncodeKey` function in Pump to speed up querying the transaction status
  - Support compressing communication data among components to reduce network resource consumption
  - Add the Arbiter tool that supports reading binlog from Kafka and replicate the data into MySQL
  - Support filtering out files that don't require replication via Reparo
  - Support replicating generated columns
  - Add the `syncer.sql-mode` configuration item to support using different sql-modes to parse DDL queries
  - Add the `syncer.ignore-table` configuration item to support filtering tables not to be replicated

- sync-diff-inspector

  - Support checkpoint to record verification status and continue the verification from last saved point after restarting
  - Add the `only-use-checksum` configuration item to check data consistency by calculating checksum
  - Support using TiDB statistics and multiple columns to split chunks for comparison to adapt to more scenarios

9.2.21.6 TiDB Ansible

- Upgrade the following monitoring components to a stable version:

  - Prometheus from V2.2.1 to V2.8.1
  - Pushgateway from V0.4.0 to V0.7.0
  - Node_exporter from V0.15.2 to V0.17.0
  - Alertmanager from V0.14.0 to V0.17.0
  - Grafana from V4.6.3 to V6.1.6
  - Ansible from V2.5.14 to V2.7.11

- Add the TiKV summary monitoring dashboard to view cluster status conveniently
- Add the TiKV trouble_shooting monitoring dashboard to remove duplicate items and facilitate troubleshooting
- Add the TiKV details monitoring dashboard to facilitate debugging and troubleshooting
- Add concurrent check for version consistency during rolling updates to improve the update performance
- Support deployment and operations for TiDB Lightning
- Optimize the `table-regions.py` script to support displaying Leader distribution by tables
- Optimize TiDB monitoring and add latency related monitoring items by SQL categories
- Modify the operating system version limit to only support the CentOS 7.0+ and Red Hat 7.0+ operating systems
- Add the monitoring item to predict the maximum QPS of the cluster (hidden by default)

9.2.22 TiDB 3.0.0-rc.3 Release Notes

Release date: June 21, 2019

TiDB version: 3.0.0-rc.3

TiDB Ansible version: 3.0.0-rc.3

9.2.22.1 Overview

On June 21, 2019, TiDB 3.0.0-rc.3 is released. The corresponding TiDB Ansible version is 3.0.0-rc.3. Compared with TiDB 3.0.0-rc.2, this release has greatly improved the stability, usability, features, the SQL optimizer, statistics, and the execution engine.

9.2.22.2 TiDB

- SQL Optimizer
  - Remove the feature of collecting virtual generated column statistics #10629
  - Fix the issue that the primary key constant overflows during point queries #10699
  - Fix the issue that using uninitialized information in `fast analyze` causes panic #10691
  - Fix the issue that executing the `create view` statement using `prepare` causes panic because of wrong column information #10713
  - Fix the issue that the column information is not cloned when handling window functions #10720
  - Fix the wrong estimation for the selectivity rate of the inner table selection in index join #10854
  - Support automatic loading statistics when the `stats-lease` variable value is 0 #10811

- Execution Engine
  - Fix the issue that resources are not correctly released when calling the `Close` function in `StreamAggExec` #10636
  - Fix the issue that the order of `table_option` and `partition_options` is incorrect in the result of executing the `show create table` statement for partitioned tables #10689
  - Improve the performance of `admin show ddl jobs` by supporting scanning data in reverse order #10687
  - Fix the issue that the result of the `show grants` statement in RBAC is incompatible with that of MySQL when this statement has the `current_user` field #10684
  - Fix the issue that UUIDs might generate duplicate values on multiple nodes #10712
  - Fix the issue that the `show view` privilege is not considered in `explain` #10635
  - Add the `split table region` statement to manually split the table Region to alleviate the hotspot issue #10765
  - Add the `split index region` statement to manually split the index Region to alleviate the hotspot issue #10764
  - Fix the incorrect execution issue when you execute multiple statements such as `create user`, `grant`, or `revoke` consecutively #10737
  - Add a blocklist to prohibit pushing down expressions to Coprocessor #10791
  - Add the feature of printing the `expensive query` log when a query exceeds the memory configuration limit #10849
  - Add the `bind-info-lease` configuration item to control the update time of the modified binding execution plan #10727
  - Fix the OOM issue in high concurrent scenarios caused by the failure to quickly release Coprocessor resources, resulted from the `execdetails.ExecDetails` pointer #10832
  - Fix the panic issue caused by the `kill` statement in some cases #10876
- Server
  - Fix the issue that goroutine might leak when repairing GC #10683
  - Support displaying the `host` information in slow queries #10693
  - Support reusing idle links that interact with TiKV #10632
  - Fix the support for enabling the `skip-grant-table` option in RBAC #10738
  - Fix the issue that `pessimistic-txn` configuration goes invalid #10825
  - Fix the issue that the actively canceled ticlient requests are still retried #10850
  - Improve performance in the case where pessimistic transactions conflict with optimistic transactions #10881
- DDL
  - Fix the issue that modifying charset using `alter table` causes the `blob` type change #10698
  - Add a feature to use `SHARD_ROW_ID_BITS` to scatter row IDs when the column contains an `AUTO_INCREMENT` attribute to alleviate the hotspot issue #10794

  - Prohibit adding stored generated columns by using the `alter table` statement #10808
  - Optimize the invalid survival time of DDL metadata to shorten the period during which the DDL operation is slower after cluster upgrade #10795

9.2.22.3 PD

- Add the `enable-two-way-merge` configuration item to allow only one-way merging #1583
- Add scheduling operations for `AddLightLearner` and `AddLightPeer` to make Region Scatter scheduling unrestricted by the limit mechanism #1563
- Fix the issue of insufficient reliability because the data might only have one replica replication when the system is started #1581
- Optimize configuration check logic to avoid configuration item errors #1585
- Adjust the definition of the `store-balance-rate` configuration to the upper limit of the number of balance operators generated per minute #1591
- Fix the issue that the store might have been unable to generate scheduled operations #1590

9.2.22.4 TiKV

- Engine
  - Fix the issue that incomplete snapshots are generated in the system caused by the iterator not checking the status #4936
  - Fix the data loss issue caused by a delay of flushing data to the disk when receiving snapshots after a power failure in abnormal conditions #4850
- Server
  - Add a feature to check the validity of the `block-size` configuration #4928
  - Add `READ_INDEX`-related monitoring metrics #4830
  - Add GC worker-related monitoring metrics #4922
- Raftstore
  - Fix the issue that the cache of the local reader is not cleared correctly #4778
  - Fix the issue that the request delay might be increased when transferring the leader and changing `conf` #4734
  - Fix the issue that a stale command is wrongly reported #4682
  - Fix the issue that the command might be pending for a long time #4810
  - Fix the issue that files are damaged after a power failure, which is caused by a delay of synchronizing the snapshot file to the disk #4807, #4850
- Coprocessor
  - Support Top-N in vector calculation #4827

  - Support `Stream` aggregation in vector calculation #4786
  - Support the `AVG` aggregate function in vector calculation #4777
  - Support the `First` aggregate function in vector calculation #4771
  - Support the `SUM` aggregate function in vector calculation #4797
  - Support the `MAX/MIN` aggregate function in vector calculation #4837
  - Support the `Like` expression in vector calculation #4747
  - Support the `MultiplyDecimal` expression in vector calculation #4849
  - Support the `BitAnd/BitOr/BitXor` expression in vector calculation #4724
  - Support the `UnaryNot` expression in vector calculation #4808

- Transaction

  - Fix the issue that an error occurs caused by non-pessimistic locking conflicts in pessimistic transactions #4801, #4883
  - Reduce unnecessary calculation for optimistic transactions after enabling pessimistic transactions to improve the performance #4813
  - Add a feature of single statement rollback to ensure that the whole transaction does not need a rollback operation in a deadlock situation #4848
  - Add pessimistic transaction-related monitoring items #4852
  - Support using the `ResolveLockLite` command to resolve lightweight locks to improve the performance when severe conflicts exist #4882

- tikv-ctl

  - Add the `bad-regions` command to support checking more abnormal conditions #4862
  - Add a feature of forcibly executing the `tombstone` command #4862

- Misc

  - Add the `dist_release` compiling command #4841

9.2.22.5 Tools

- TiDB Binlog

  - Fix the wrong offset issue caused by Pump not checking the returned value when it fails to write data #640
  - Add the `advertise-addr` configuration in Drainer to support the bridge mode in the container environment #634
  - Add the `GetMvccByEncodeKey` function in Pump to speed up querying the transaction status #632

9.2.22.6 TiDB Ansible

- Add a monitoring item to predict the maximum QPS value of the cluster ("hide" by default) #f5cfa4d

9.2.23 TiDB 3.0.0-rc.2 Release Notes

Release date: May 28, 2019

TiDB version: 3.0.0-rc.2

TiDB Ansible version: 3.0.0-rc.2

9.2.23.1 Overview

On May 28, 2019, TiDB 3.0.0-rc.2 is released. The corresponding TiDB Ansible version is 3.0.0-rc.2. Compared with TiDB 3.0.0-rc.1, this release has greatly improved the stability, usability, features, the SQL optimizer, statistics, and the execution engine.

9.2.23.2 TiDB

- SQL Optimizer
  - Support Index Join in more scenarios #10540
  - Support exporting historical statistics #10291
  - Support the incremental `Analyze` operation on monotonically increasing index columns #10355
  - Neglect the NULL value in the `Order By` clause #10488
  - Fix the wrong schema information calculation of the `UnionAll` logical operator when simplifying the column information #10384
  - Avoid modifying the original expression when pushing down the `Not` operator #10363
  - Support the `dump/load` correlation of histograms #10573
- Execution Engine
  - Handle virtual columns with a unique index properly when fetching duplicate rows in `batchChecker` #10370
  - Fix the scanning range calculation issue for the `CHAR` column #10124
  - Fix the issue of `PointGet` incorrectly processing negative numbers #10113
  - Merge `Window` functions with the same name to improve execution efficiency #9866
  - Allow the `RANGE` frame in a `Window` function to contain no `OrderBy` clause #10496
- Server
  - Fix the issue that TiDB continuously creates a new connection to TiKV when a fault occurs in TiKV #10301
  - Make `tidb_disable_txn_auto_retry` affect all retryable errors instead of only write conflict errors #10339
  - Allow DDL statements without parameters to be executed using `prepare` ↪ `/execute` #10144
  - Add the `tidb_back_off_weight` variable to control the backoff time #10266

  - Prohibit TiDB retrying non-automatically committed transactions in default conditions by setting the default value of `tidb_disable_txn_auto_retry` to `on` #10266
  - Fix the database privilege judgment of `role` in `RBAC` #10261
  - Support the pessimistic transaction model (experimental) #10297
  - Reduce the wait time for handling lock conflicts in some cases #10006
  - Make the Region cache able to visit follower nodes when a fault occurs in the leader node #10256
  - Add the `tidb_low_resolution_tso` variable to control the number of TSOs obtained in batches and reduce the times of transactions obtaining TSO to adapt for scenarios where data consistency is not so strictly required #10428

- DDL

  - Fix the uppercase issue of the charset name in the storage of the old version of TiDB #10272
  - Support `preSplit` of table partition, which pre-allocates table Regions when creating a table to avoid write hotspots after the table is created #10221
  - Fix the issue that TiDB incorrectly updates the version information in PD in some cases #10324
  - Support modifying the charset and collation using the `ALTER DATABASE` statement #10393
  - Support splitting Regions based on the index and range of the specified table to relieve hotspot issues #10203
  - Prohibit modifying the precision of the decimal column using the `alter table` statement #10433
  - Fix the restriction for expressions and functions in hash partition #10273
  - Fix the issue that adding indexes in a table that contains partitions will in some cases cause TiDB panic #10475
  - Validate table information before executing the DDL to avoid invalid table schemas #10464
  - Enable hash partition by default; and enable range columns partition when there is only one column in the partition definition #9936

9.2.23.3 PD

- Enable the Region storage by default to store the Region metadata #1524
- Fix the issue that hot Region scheduling is preempted by another scheduler #1522
- Fix the issue that the priority for the leader does not take effect #1533
- Add the gRPC interface for `ScanRegions` #1535
- Push operators actively #1536
- Add the store limit mechanism for separately controlling the speed of operators for each store #1474
- Fix the issue of inconsistent `Config` status #1476

9.2.23.4 TiKV

- Engine
  - Support multiple column families sharing a block cache #4563
- Server
  - Remove `TxnScheduler` #4098
  - Support pessimistic lock transactions #4698
- Raftstore
  - Support hibernate Regions to reduce the consumption of the raftstore CPU #4591
  - Fix the issue that the leader does not reply to the `ReadIndex` requests for the learner #4653
  - Fix transferring leader failures in some cases #4684
  - Fix the dirty read issue in some cases #4688
  - Fix the issue that a snapshot may lose applied data in some cases #4716
- Coprocessor
  - Add more RPN functions
    * `LogicalOr` #4691
    * `LTReal` #4602
    * `LEReal` #4602
    * `GTReal` #4602
    * `GEReal` #4602
    * `NEReal` #4602
    * `EQReal` #4602
    * `IsNull` #4720
    * `IsTrue` #4720
    * `IsFalse` #4720
    * Support comparison arithmetic for `Int` #4625
    * Support comparison arithmetic for `Decimal` #4625
    * Support comparison arithmetic for `String` #4625
    * Support comparison arithmetic for `Time` #4625
    * Support comparison arithmetic for `Duration` #4625
    * Support comparison arithmetic for `Json` #4625
    * Support plus arithmetic for `Int` #4733
    * Support plus arithmetic for `Real` #4733
    * Support plus arithmetic for `Decimal` #4733
    * Support MOD functions for `Int` #4727
    * Support MOD functions for `Real` #4727
    * Support MOD functions for `Decimal` #4727
    * Support minus arithmetic for `Int` #4746
    * Support minus arithmetic for `Real` #4746
    * Support minus arithmetic for `Decimal` #4746

9.2.23.5 Tools

- TiDB Binlog
  - Add a metric to track the delay of data replication downstream #594
- TiDB Lightning
  - Support merging sharded databases and tables #95
  - Add the retry mechanism for KV write failure #176
  - Update the default value of `table-concurrency` to 6 #175
  - Reduce required configuration items by automatically discovering `tidb.pd-addr` and `tidb.port` if they are not provided #173

9.2.24 TiDB 3.0.0-rc.1 Release Notes

Release Date: May 10, 2019

TiDB version: 3.0.0-rc.1

TiDB Ansible version: 3.0.0-rc.1

9.2.24.1 Overview

On May 10, 2019, TiDB 3.0.0-rc.1 is released. The corresponding TiDB Ansible version is 3.0.0-rc.1. Compared with TiDB 3.0.0-beta.1, this release has greatly improved the stability, usability, features, the SQL optimizer, statistics, and the execution engine.

9.2.24.2 TiDB

- SQL Optimizer
  - Improve the accuracy of cost estimates by using order correlation between columns; introduce a heuristic parameter `tidb_opt_correlation_exp_factor` to control the preference for index scans for scenarios when correlation cannot be directly used for estimation. #9839
  - Match more prefix columns of the indexes when extracting access conditions of composite indexes if there are relevant columns in the filter #10053
  - Use the dynamic programming algorithm to specify the execution order of join operations when the number of tables participating in the join is less than the value of `tidb_opt_join_reorder_threshold`. #8816
  - Match more prefix columns of the indexes in the inner tables that build the index join when using composite indexes as the access conditions #8471
  - Improve the accuracy of row count estimation for single-column indexes with NULL values #9474

  - Specially handle `GROUP_CONCAT` when eliminating aggregate functions during the logical optimization phase to prevent incorrect executions #9967
  - Properly push the filter down to child nodes of the join operator if the filter is a constant #9848
  - Specially handle some functions such as `RAND()` when pruning columns during the logical optimization phase to prevent incompatibilities with MySQL #10064
  - Support `FAST ANALYZE`, which speeds up statistics collection by sampling the region instead of scanning the entire region. This feature is controlled by the variable `tidb_enable_fast_analyze`. #10258
  - Support SQL Plan Management, which ensures execution stability by performing execution plan binding for SQL statements. This feature is currently in beta and only supports bound execution plans for SELECT statements. It is not recommended to use it in the production environment. #10284

- Execution Engine

  - Support tracking and controlling memory usage in three operators - `TableReader`, `IndexReader` and `IndexLookupReader` #10003
  - Support showing more information about coprocessor tasks in the slow log such as the number of tasks in coprocessor, the average/longest/90% of execution/waiting time and the addresses of the TiKVs which take the longest execution time or waiting time #10165
  - Support the prepared DDL statements with no placeholders #10144

- Server

  - Only allow the DDL owner to execute bootstrap when TiDB is started #10029
  - Add the variable `tidb_skip_isolation_level_check` to prevent TiDB from reporting errors when setting the transaction isolation level to SERIALIZABLE #10065
  - Merge the implicit commit time and the SQL execution time in the slow log #10294
    * Support for SQL Roles (RBAC Privilege Management)
    * Support `SHOW GRANT` #10016
    * Support `SET DEFAULT ROLE` #9949
  - Support `GRANT ROLE` #9721
  - Fix the `ConnectionEvent` error from the `whitelist` plugin that makes TiDB exit #9889
  - Fix the issue of mistakenly adding read-only statements to the transaction history #9723
  - Improve `kill` statements to stop SQL execution and release resources more quickly #9844
  - Add a startup option `config-check` to check the validity of the configuration file #9855
  - Fix the validity check of inserting NULL fields when the strict SQL mode is disabled #10161

- DDL
  - Add the `pre_split_regions` option for `CREATE TABLE` statements; this option supports pre-splitting the Table Region when creating a table to avoid write hot spots caused by lots of writes after the table creation #10138
  - Optimize the execution performance of some DDL statements #10170
  - Add the warning that full-text indexes are not supported for `FULLTEXT KEY` #9821
  - Fix the compatibility issue for the UTF8 and UTF8MB4 charsets in the old versions of TiDB #9820
  - Fix the potential bug in `shard_row_id_bits` of a table #9868
  - Fix the bug that the column charset is not changed after the table charset is changed #9790
  - Fix a potential bug in `SHOW COLUMN` when using `BINARY/BIT` as the column default value #9897
  - Fix the compatibility issue in displaying `CHARSET/COLLATION` descriptions in the `SHOW FULL COLUMNS` statement #10007
  - Fix the issue that the `SHOW COLLATIONS` statement only lists collations supported by TiDB #10186

9.2.24.3 PD

- Upgrade ETCD #1452
  - Unify the log format of etcd and PD server
  - Fix the issue of failing to elect Leader by PreVote
  - Support fast dropping the "propose" and "read" requests that are to fail to avoid blocking the subsequent requests
  - Fix the deadlock issue of Lease
- Fix the issue that a hot store makes incorrect statistics of keys #1487
- Support forcibly rebuilding a PD cluster from a single PD node #1485
- Fix the issue that `regionScatterer` might generate an invalid `OperatorStep` #1482
- Fix the too short timeout issue of the `MergeRegion` operator #1495
- Support giving high priority to hot region scheduling #1492
- Add the metrics for recording the time of handling TSO requests on the PD server side #1502
- Add the corresponding Store ID and Address to the metrics related to the store #1506
- Support the `GetOperator` service #1477
- Fix the issue that the error cannot be sent in the Heartbeat stream because the store cannot be found #1521

9.2.24.4 TiKV

- Engine
  - FIx the issue that may cause incorrect statistics on read traffic #4436

  - Fix the issue that may cause prefix extractor panic when deleting a range #4503
  - Optimize memory management to reduce memory allocation and copying for `Iterator Key Bound Option` #4537
  - Fix the issue that failing to consider learner log gap may in some cases cause panic #4559
  - Support `block cache` sharing among different `column families` #4612

- Server
  - Reduce context switch overhead of `batch commands` #4473
  - Check the validity of seek iterator status #4470

- RaftStore
  - Support configurable `properties index distance` #4517

- Coprocessor
  - Add batch index scan executor #4419
  - Add vectorized evaluation framework #4322
  - Add execution summary framework for batch executors #4433
  - Check the maximum column when constructing the RPN expression to avoid invalid column offset that may cause evaluation panic #4481
  - Add `BatchLimitExecutor` #4469
  - Replace the original `futures-cpupool` with `tokio-threadpool` in ReadPool to reduce context switch #4486
  - Add batch aggregation framework #4533
  - Add `BatchSelectionExecutor` #4562
  - Add batch aggression function `AVG` #4570
  - Add RPN function `LogicalAnd`#4575

- Misc
  - Support `tcmalloc` as a memory allocator #4370

9.2.24.5 Tools

- TiDB Binlog
  - Fix the replication abortion issue when binlog data for the primary key column of unsigned int type is negative #573
  - Provide no compression option when downstream is `pb`; modify the downstream name from `pb` to `file` #559
  - Add the `storage.sync-log` configuration item in Pump that allows asynchronous flush on local storage #509
  - Support traffic compression for communications between Pump and Drainer #495
  - Add the `syncer.sql-mode` configuration item in Drainer to support parsing DDL queries in different sql-mode #511

  - Add the `syncer.ignore-table` configuration item to support filtering out tables that do not require replication #520

- Lightning
  - Use row IDs or default column values to populate the column data missed in the dump file #170
  - Fix the bug in Importer that import success may still be returned even if part of the SST failed to be imported #4566
  - Support speed limit in Importer when uploading SST to TiKV #4412
  - Support importing tables by size to reduce impacts on the cluster brought by Checksum and Analyze for big tables, and improve the success rate for Checksum and Analyze #156
  - Improve Lightning's SQL encoding performance by 50% by directly parsing data source file as types.Datum of TiDB and saving extra parsing overhead from the KV encoder #145
  - Change log format to Unified Log Format #162
  - Add some command line options for use when the configuration file is missing #157
- sync-diff-inspector
  - Support checkpoint to record verification status and continue the verification from last saved point after restarting #224
  - Add the `only-use-checksum` configuration item to check data consistency by calculating checksum #215

9.2.24.6 TiDB Ansible

- Support more TiKV monitoring panels and update versions for Ansible, Grafana, and Prometheus #727
  - Summary dashboard for viewing cluster status
  - trouble_shooting dashboard for troubleshooting issues
  - Details dashboard for developers to analyze issues
- Fix the bug that causes the downloading failure of TiDB Binlog of Kafka version #730
- Modify version limits on supported operating systems as CentOS 7.0+ and later, and Red Hat 7.0 and later #733
- Change version detection mode during the rolling update to multi-concurrent #736
- Update documentation links in README #740
- Remove redundant TiKV monitoring metrics; add new metrics for troubleshooting #735
- Optimize `table-regions.py` script to display leader distribution by table #739
- Update configuration file for Drainer #745
- Optimize TiDB monitoring with new panels that display latencies by SQL categories #747
- Update the Lightning configuration file and add the `tidb_lightning_ctl` script #1e946f8

9.2.25 TiDB 3.0.0 Beta.1 Release Notes

Release Date: March 26, 2019

TiDB version: 3.0.0-beta.1

TiDB Ansible version: 3.0.0-beta.1

9.2.25.1 Overview

On March 26, 2019, TiDB 3.0.0 Beta.1 is released. The corresponding TiDB Ansible version is 3.0.0 Beta.1. Compared with TiDB 3.0.0 Beta, this release has greatly improved the stability, usability, features, the SQL optimizer, statistics, and the execution engine.

9.2.25.2 TiDB

- SQL Optimizer
  - Support calculating the Cartesian product by using `Sort Merge Join` #9032
  - Support Skyline Pruning, with some rules to prevent the execution plan from relying too heavily on statistics #9337
  - Support Window Functions
    * `NTILE` #9682
    * `LEAD` and `LAG` #9672
    * `PERCENT_RANK` #9671
    * `NTH_VALUE` #9596
    * `CUME_DIST` #9619
    * `FIRST_VALUE` and `LAST_VALUE` #9560
    * `RANK` and `DENSE_RANK` #9500
    * `RANGE FRAMED` #9450
    * `ROW FRAMED` #9358
    * `ROW NUMBER` #9098
  - Add a type of statistic that indicates the order correlation between columns and the handle column #9315
- SQL Execution Engine
  - Add built-in functions
    * `JSON_QUOTE` #7832
    * `JSON_ARRAY_APPEND` #9609
    * `JSON_MERGE_PRESERVE` #8931
    * `BENCHMARK` #9252
    * `COALESCE` #9087
    * `NAME_CONST` #9261
  - Optimize the Chunk size based on the query context, to reduce the execution time of SQL statements and resources consumption of the cluster #6489

- Privilege management
  - Support `SET ROLE` and `CURRENT_ROLE` #9581
  - Support `DROP ROLE` #9616
  - Support `CREATE ROLE` #9461
- Server
  - Add the `/debug/zip` HTTP interface to get information of the current TiDB instance #9651
  - Support the `show pump status` and `show drainer status` SQL statements to check the Pump or Drainer status 9456
  - Support modifying the Pump or Drainer status by using SQL statements #9789
  - Support adding HASH fingerprints to SQL text for easy tracking of slow SQL statements #9662
  - Add the `log_bin` system variable ("0" by default) to control the enabling state of binlog; only support checking the state currently #9343
  - Support managing the sending binlog strategy by using the configuration file #9864
  - Support querying the slow log by using the `INFORMATION_SCHEMA.SLOW_QUERY` memory table #9290
  - Change the MySQL version displayed in TiDB from 5.7.10 to 5.7.25 #9553
  - Unify the log format for easy collection and analysis by tools
  - Add the `high_error_rate_feedback_total` monitoring item to record the difference between the actual data volume and the estimated data volume based on statistics #9209
  - Add the QPS monitoring item in the database dimension, which can be enabled by using a configuration item #9151
- DDL
  - Add the `ddl_error_count_limit` global variable ("512" by default) to limit the number of DDL task retries (If this number exceeds the limit, the DDL task is canceled) #9295
  - Support ALTER ALGORITHM `INPLACE`/`INSTANT` #8811
  - Support the `SHOW CREATE VIEW` statement #9309
  - Support the `SHOW CREATE USER` statement #9240

9.2.25.3 PD

- Unify the log format for easy collection and analysis by tools
- Simulator
  - Support different heartbeat intervals in different stores #1418
  - Add a case about importing data #1263
- Make hotspot scheduling configurable #1412

- Add the store address as the dimension monitoring item to replace the previous Store ID #1429
- Optimize the `GetStores` overhead to speed up the Region inspection cycle #1410
- Add an interface to delete the Tombstone Store #1472

9.2.25.4 TiKV

- Optimize the Coprocessor calculation execution framework and implement the TableScan section, with the Single TableScan performance improved by 5% ~ 30%
  - Implement the definition of the `BatchRows` row and the `BatchColumn` column #3660
  - Implement `VectorLike` to support accessing encoded and decoded data in the same way #4242
  - Define the `BatchExecutor` to interface and implement the way of converting requests to `BatchExecutor` #4243
  - Implement transforming the expression tree into the RPN format #4329
  - Implement the `BatchTableScanExecutor` vectorization operator to accelerate calculation #4351
- Unify the log format for easy collection and analysis by tools
- Support using the Local Reader to read in the Raw Read interface #4222
- Add metrics about configuration information #4206
- Add metrics about key exceeding bound #4255
- Add an option to control panic or return an error when encountering the key exceeding bound error #4254
- Add support for the `INSERT` operation, make prewrite succeed only when keys do not exist, and eliminate `Batch Get` #4085
- Use more fair batch strategy in the Batch System #4200
- Support Raw scan in tikv-ctl #3825

9.2.25.5 Tools

- TiDB Binlog
  - Add the Arbiter tool that supports reading binlog from Kafka and replicate the data into MySQL
  - Support filtering files that do not need to be replicated
  - Support replicating generated columns
- Lightning
  - Support disabling TiKV periodic Level-1 compaction, and when the TiKV cluster version is 2.1.4 or later, Level-1 compaction is automatically executed in the import mode #119, #4199

  - Add the `table_concurrency` configuration item to limit the number of import engines ("16" by default) and avoid overusing the importer disk space #119
  - Support saving the intermediate state SST to the disk, to reduce memory usage #4369
  - Optimize the import performance of TiKV-Importer and support separate import of data and indexes for large tables #132
  - Support importing CSV files #111

- Data replication comparison tool (sync-diff-inspector)

  - Support using TiDB statistics to split chunks to be compared #197
  - Support using multiple columns to split chunks to be compared #197

9.2.26 TiDB 3.0 Beta Release Notes

On January 19, 2019, TiDB 3.0 Beta is released. The corresponding TiDB Ansible 3.0 Beta is also released. TiDB 3.0 Beta builds on TiDB 2.1 with an added focus in stability, the SQL optimizer, statistics, and the execution engine.

9.2.26.1 TiDB

- New Features

  - Support Views
  - Support Window Functions
  - Support Range Partitioning
  - Support Hash Partitioning

- SQL Optimizer

  - Re-support the optimization rule of `AggregationElimination` #7676
  - Optimize the `NOT EXISTS` subquery and convert it to Anti Semi Join #7842
  - Add the `tidb_enable_cascades_planner` variable to support the new Cascades optimizer. Currently, the Cascades optimizer is not yet fully implemented and is turned off by default #7879
  - Support using Index Join in transactions #7877
  - Optimize the constant propagation on the Outer Join, so that the filtering conditions related to the Outer table in the Join result can be pushed down through the Outer Join to the Outer table, reducing the useless calculation of the Outer Join and improving the execution performance #7794
  - Adjust the optimization rule of Projection Elimination to the position after the Aggregation Elimination, to avoid redundant `Project` operators #7909
  - Optimize the `IFNULL` function and eliminate this function when the input parameter has a non-NULL attribute #7924
  - Support Range for `_tidb_rowid` construction queries, to avoid full table scan and reduce cluster stress #8047

  - Optimize the `IN` subquery to do the Inner Join after the aggregation, and add the `tidb_opt_insubq_to_join_and_agg` variable to control whether to enable this optimization rule and open it by default #7531
  - Support using subqueries in the `DO` statement #8343
  - Add the optimization rule of Outer Join elimination to reduce unnecessary table scan and Join operations and improve execution performance #8021
  - Modify the Hint behavior of the `TIDB_INLJ` optimizer, and the optimizer will use the table specified in Hint as the Inner table of Index Join #8243
  - Use `PointGet` in a wide range so that it can be used when the execution plan cache of the `Prepare` statement takes effect #8108
  - Introduce the greedy `Join Reorder` algorithm to optimize the join order selection when joining multiple tables #8394
  - Support View #8757
  - Support Window Function #8630
  - Return warning to the client when `TIDB_INLJ` is not in effect, to enhance usability #9037
  - Support deducing the statistics for filtered data based on filtering conditions and table statistics #7921
  - Improve the Partition Pruning optimization rule of Range Partition #8885

- SQL Executor

  - Optimize the `Merge Join` operator to support the empty `ON` condition #9037
  - Optimize the log and print the user variables used when executing the `EXECUTE` statement #7684
  - Optimize the log to print slow query information for the `COMMIT` statement #7951
  - Support the `EXPLAIN ANALYZE` feature to make the SQL tuning process easier #7827
  - Optimize the write performance of wide tables with many columns #7935
  - Support `admin show next_row_id` #8242
  - Add the `tidb_init_chunk_size` variable to control the size of the initial Chunk used by the execution engine #8480
  - Improve `shard_row_id_bits` and cross-check the auto-increment ID #8936

- `Prepare` Statement

  - Prohibit adding the `Prepare` statement containing subqueries to the query plan cache to guarantee the query plan is correct when different user variables are input #8064
  - Optimize the query plan cache to guarantee the plan can be cached when the statement contains non-deterministic functions #8105
  - Optimize the query plan cache to guarantee the query plan of `DELETE`/`UPDATE` ↪ `/INSERT` can be cached #8107
  - Optimize the query plan cache to remove the corresponding plan when executing the `DEALLOCATE` statement #8332
  - Optimize the query plan cache to avoid the TiDB OOM issue caused by caching too many plans by limiting the memory usage #8339

  - Optimize the `Prepare` statement to support using the `?` placeholder in the `ORDER BY/GROUP BY/LIMIT` clause #8206

- Privilege Management

  - Add the privilege check for the `ANALYZE` statement #8486
  - Add the privilege check for the `USE` statement #8414
  - Add the privilege check for the `SET GLOBAL` statement #8837
  - Add the privilege check for the `SHOW PROCESSLIST` statement #7858

- Server

  - Support the `Trace` feature #9029
  - Support the plugin framework #8788
  - Support using `unix_socket` and TCP simultaneously to connect to the database #8836
  - Support the `interactive_timeout` system variable #8573
  - Support the `wait_timeout` system variable #8346
  - Support splitting a transaction into multiple transactions based on the number of statements using the `tidb_batch_commit` variable #8293
  - Support using the `ADMIN SHOW SLOW` statement to check slow logs #7785

- Compatibility

  - Support the `ALLOW_INVALID_DATES` SQL mode #9027
  - Improve `LoadData` fault-tolerance for the CSV file #9005
  - Support the MySQL 320 handshake protocol #8812
  - Support using the unsigned `bigint` column as the auto-increment column #8181
  - Support the `SHOW CREATE DATABASE IF NOT EXISTS` syntax #8926
  - Abandon the predicate pushdown operation when the filtering condition contains a user variable to improve the compatibility with MySQL's behavior of using user variables to mock the Window Function behavior #8412

- DDL

  - Support fast recovery of mistakenly deleted tables #7937
  - Support adjusting the number of concurrencies of `ADD INDEX` dynamically #8295
  - Support changing the character set of tables or columns to `utf8`/`utf8mb4` #8037
  - Change the default character set from `utf8` to `utf8mb4` #7965
  - Support Range Partition #8011

9.2.26.2 Tools

- TiDB Lightning

  - Speed up converting SQL statements to KV pairs remarkably #110
  - Support batch import for a single table to improve import performance and stability #113

9.2.26.3 PD

- Add `RegionStorage` to store Region metadata separately #1237
- Add shuffle hot Region scheduler #1361
- Add scheduling parameter related metrics #1406
- Add cluster label related metrics #1402
- Add the importing data simulator #1263
- Fix the `Watch` issue about leader election #1396

9.2.26.4 TiKV

- Support distributed GC #3179
- Check RocksDB Level 0 files before applying snapshots to avoid Write Stall #3606
- Support reverse `raw_scan` and `raw_batch_scan` #3742
- Support using HTTP to obtain monitoring information #3855
- Support DST better #3786
- Support receiving and sending Raft messages in batch #3931
- Introduce a new storage engine Titan #3985
- Upgrade gRPC to v1.17.2 #4023
- Support receiving the client requests and sending replies in batch #4043
- Support multi-thread Apply #4044
- Support multi-thread Raftstore #4066

9.3 v2.1

9.3.1 TiDB 2.1.19 Release Notes

Release date: December 27, 2019

TiDB version: 2.1.19

TiDB Ansible version: 2.1.19

9.3.1.1 TiDB

- SQL Optimizer
  - Optimize the scenario of `select max(_tidb_rowid)from t` to avoid full table scan #13294
  - Fix the incorrect results caused by the incorrect value assigned to the user variable in the query and the push-down of predicates #13230
  - Fix the issue that the statistics are not accurate because a data race occurs when statistics are updated #13690

  - Fix the issue that the result is incorrect when the `UPDATE` statement contains both a sub-query and a stored generated column; fix the `UPDATE` statement execution error when this statement contains two same-named tables from different databases #13357
  - Fix the issue that the query plan might be incorrectly selected because the `PhysicalUnionScan` operator incorrectly sets the statistics #14134
  - Remove the `minAutoAnalyzeRatio` constraint to make the automatic `ANALYZE` more timely #14013
  - Fix the issue that the estimated number of rows is greater than `1` when the `WHERE` clause contains an equal condition on the unique key #13385

- SQL Execution Engine

  - Fix the precision overflow when using `int64` as the intermediate result of `unit64` in `ConvertJSONToInt` #13036
  - Fix the issue that when the `SLEEP` function is in a query (for example, `select 1 from (select sleep(1))t;`), column pruning causes `sleep(1)` in the query to be invalid #13039
  - Reduce memory overhead by reusing `Chunk` in the `INSERT ON DUPLICATE UPDATE` statement #12999
  - Add more transaction-related fields for the `slow_query` table #13129:
    * `Prewrite_time`
    * `Commit_time`
    * `Get_commit_ts_time`
    * `Commit_backoff_time`
    * `Backoff_types`
    * `Resolve_lock_time`
    * `Local_latch_wait_time`
    * `Write_key`
    * `Write_size`
    * `Prewrite_region`
    * `Txn_retry`
  - Fix the issue that a subquery contained in an `UPDATE` statement is incorrectly converted; fix the `UPDATE` execution failure when the `WHERE` clause contains a subquery #13120
  - Support executing `ADMIN CHECK TABLE` on partitioned tables #13143
  - Fix the issue that the precision of statements such as `SHOW CREATE TABLE` is incomplete when `ON UPDATE CURRENT_TIMESTAMP` is used as a column attribute and floating point precision is specified #12462
  - Fix the panic occurred when executing the `SELECT * FROM information_schema.KEY_COLUMN_USAGE` statement because the foreign key is not checked when dropping, modifying or changing the column #14162
  - Fix the issue that the returned data might be duplicated when `Streaming` is enabled in TiDB #13255
  - Fix the `Invalid time format` error caused by daylight saving time #13624

  - Fix the issue that data is incorrect because the precision is lost when an integer is converted to an unsigned floating point or decimal type #13756
  - Fix the issue that an incorrect type of value is returned when the `Quote` function handles the `NULL` value #13681
  - Fix the issue that the timezone is incorrect after parsing the date from strings using `gotime.Local` #13792
  - Fix the issue that the result might be incorrect because the `binSearch` function does not return an error in the implementation of `builtinIntervalRealSig` #13768
  - Fix the issue that an error might occur when converting the string type to the floating point type in the `INSERT` statement execution #14009
  - Fix the incorrect result returned from the `sum(distinct)` function #13041
  - Fix the issue that `data too long` is returned when `CAST` converting the data in `union` of the same location to the merged type because the returned type length of the `jsonUnquoteFunction` function is given an incorrect value #13645
  - Fix the issue that the password cannot be set because the privilege check is too strict #13805

- Server

  - Fix the issue that `KILL CONNECTION` might cause the goroutine leak #13252
  - Support getting the binlog status of all TiDB nodes via the `info/all` interface of the HTTP API #13188
  - Fix the failure to build the TiDB project on Windows #13650
  - Add the `server-version` configuration item to control and modify the version of TiDB server #13904
  - Fix the issue that the binary `plugin` compiled with Go1.13 does not run normally #13527

- DDL

  - Use the table's `COLLATE` instead of the system's default charset in the column when a table is created and the table contains `COLLATE` #13190
  - Limit the length of the index name when creating a table #13311
  - Fix the issue that the length of the table name is not checked when renaming a table #13345
  - Check the width range of the `BIT` column #13511
  - Make the error information output from `change/modify column` more understandable #13798
  - Fix the issue that when executing the `drop column` operation that has not yet been handled by the downstream Drainer, the downstream might receive DML operations without the affected column #13974

9.3.1.2 TiKV

- Raftstore

  - Fix the panic occurred when restarting TiKV and `is_merging` is given an incorrect value in the process of merging Regions and applying the Compact log #5884

- Importer
  - Remove the limit on the gRPC message length #5809

9.3.1.3 PD

- Improve the performance of the HTTP API for getting all Regions #1988
- Upgrade etcd to fix the issue that etcd PreVote cannot elect a leader (downgrade not supported) #2052

9.3.1.4 Tools

- TiDB Binlog
  - Optimize the node status information output through binlogctl #777
  - Fix the panic occurred because of the `nil` value in the Drainer filter configuration #802
  - Optimize the `Graceful` exit of Pump #825
  - Add more detailed monitoring metrics when Pump writes binlog data #830
  - Optimize Drainer's logic to refresh table information after Drainer has executed a DDL operation #836
  - Fix the issue that the commit binlog of a DDL operation is ignored when Pump does not receive this binlog #855

9.3.1.5 TiDB Ansible

- Rename the `Uncommon Error OPM` monitoring item of TiDB service to `Write Binlog Error` and add the corresponding alert message #1038
- Upgrade TiSpark to 2.1.8 #1063

9.3.2 TiDB 2.1.18 Release Notes

Release date: November 4, 2019

TiDB version: 2.1.18

TiDB Ansible version: 2.1.18

9.3.2.1 TiDB

- SQL Optimizer
  - Fix the issue that invalid query ranges might appear when split by feedback #12172
  - Fix the issue that the privilege check is incorrect in point get plan #12341
  - Optimize execution performance of the `select ... limit ... offset ⋯` statement by pushing the Limit operator down to the `IndexLookUpReader` execution logic #12380
  - Support using parameters in `ORDER BY`, `GROUP BY` and `LIMIT OFFSET` #12514
  - Fix the issue that `IndexJoin` on the partition table returns incorrect results #12713
  - Fix the issue that the `str_to_date` function in TiDB returns a different result from MySQL when the date string and the format string do not match #12757
  - Fix the issue that outer join is incorrectly converted to inner join when the `cast` function is included in the query conditions #12791
  - Fix incorrect expression passing in the join condition of `AntiSemiJoin` #12800
- SQL Engine
  - Fix the incorrectly rounding of time (for example, `2019-09-11 11:17:47.999999666` ↪ should be rounded to `2019-09-11 11:17:48`) #12259
  - Fix the issue that the duration by `sql_type` for the `PREPARE` statement is not shown in the monitoring record #12329
  - Fix the panic issue when the `from_unixtime` function handles null #12572
  - Fix the compatibility issue that when an invalid value is inserted as the `YEAR` type, the result is `NULL` instead of `0000` #12744
  - Improve the behavior of the `AutoIncrement` column when it is implicitly allocated, to keep it consistent with the default mode of MySQL auto-increment locking ("consecutive" lock mode): for the implicit allocation of multiple `AutoIncrement` IDs in a single-line `Insert` statement, TiDB guarantees the continuity of the allocated values. This improvement ensures that the JDBC `getGeneratedKeys()` method will get the correct results in any scenario #12619
  - Fix the issue that the query is hanged when `HashAgg` serves as a child node of `Apply` #12769
  - Fix the issue that the `AND` and `OR` logical expressions return incorrect results when it comes to type conversion #12813
- Server
  - Fix the issue that the `SLEEP()` function is invalid for the `KILL TIDB QUERY` statements #12159
  - Fix the issue that no error is reported when `AUTO_INCREMENT` incorrectly allocates `MAX int64` and `MAX uint64` #12210
  - Fix the issue that the slow query logs are not recorded when the log level is `ERROR` #12373

  - Adjust the number of times that TiDB caches schema changes and corresponding changed table information from 100 to 1024, and support modification by using the `tidb_max_delta_schema_count` system variable #12515
  - Change the query start time from the point of "starting to execute" to "starting to compile" to make SQL statistics more accurate #12638
  - Add the record of `set session autocommit` in TiDB logs #12568
  - Record SQL query start time in `SessionVars` to prevent it from being reset during plan execution #12676
  - Support `?` placeholder in `ORDER BY`, `GROUP BY` and `LIMIT OFFSET` #12514
  - Add the `Prev_stmt` field in slow query logs to output the previous statement when the last statement is `COMMIT` #12724
  - Record the last statement before `COMMIT` into the log when the `COMMIT` fails in an explicitly committed transaction #12747
  - Optimize the saving method of the previous statement when the TiDB server executes a SQL statement to improve performance #12751
  - Fix the panic issue caused by `FLUSH PRIVILEGES` statements under the `skip-grant-table=true` configuration #12816
  - Increase the default minimum step of applying AutoID from `1000` to `30000` to avoid performance bottleneck when there are many write requests in a short time #12891
  - Fix the issue that the failed `Prepared` statement is not print in the error log when TiDB panics #12954
  - Fix the issue that the `COM_STMT_FETCH` time record in slow query logs is inconsistent with that in MySQL #12953
  - Add an error code in the error message for write conflicts to quickly locate the cause #12878

- DDL

  - Disallow dropping the `AUTO INCREMENT` attribute of a column by default. Modify the value of the `tidb_allow_remove_auto_inc` variable if you do need to drop this attribute. See TiDB Specific System Variables for more details. #12146
  - Support multiple `uniques` when creating a unique index in the `Create Table` statement #12469
  - Fix a compatibility issue that if the foreign key constraint in `CREATE TABLE` statement has no schema, schema of the created table should be used instead of returning a `No Database selected` error #12678
  - Fix the issue that the `invalid list index` error is reported when executing `ADMIN CANCEL DDL JOBS` #12681

- Monitor

  - Add types for backoff monitoring and supplement the backoff time that is not recorded before, such as the backoff time when committing #12326
  - Add a new metric to monitor `Add Index` operation progress #12389

9.3.2.2 PD

- Improve the `--help` command output of pd-ctl #1772

9.3.2.3 Tools

- TiDB Binlog
  - Fix the issue that `ALTER DATABASE` related DDL operations cause Drainer to exit abnormally #770
  - Support querying the transaction status information for Commit binlog to improve replication efficiency #761
  - Fix the issue that a Pump panic might occur when Drainer's `start_ts` is greater than Pump's largest `commit_ts` #759

9.3.2.4 TiDB Ansible

- Add two monitoring items "queue size" and "query histogram" for TiDB Binlog #952
- Update TiDB alerting rules #961
- Check the configuration file before the deployment and upgrade #973
- Add a new metric to monitor index speed in TiDB #987
- Update TiDB Binlog monitoring dashboard to make it compatible with Grafana v4.6.3 #993

9.3.3 TiDB 2.1.17 Release Notes

Release date: September 11, 2019

TiDB version: 2.1.17

TiDB Ansible version: 2.1.17

- New features
  - Add the `WHERE` clause in TiDB's `SHOW TABLE REGIONS` syntax
  - Add the `config-check` feature in TiKV and PD to check the configuration items
  - Add the `remove-tombstone` command in pd-ctl to clear tombstone store records
  - Add the `worker-count` and `txn-batch` configuration items in Reparo to control the recovery speed
- Improvements
  - Optimize PD's scheduling process by supporting actively pushing operators
  - Optimize TiKV's starting process to reduce jitters caused by restarting nodes
- Changed behaviors
  - Change `start ts` in TiDB slow query logs from the last retry time to the first execution time

- Replace the `Index_ids` field in TiDB slow query logs with the `Index_names` field to improve the usability of slow query logs
- Add the `split-region-max-num` parameter in TiDB's configuration files to modify the maximum number of Regions allowed by the `SPLIT TABLE` syntax, which is increased from 1,000 to 10,000 in the default configuration

9.3.3.1 TiDB

- SQL Optimizer
  - Fix the issue that the error message is not returned correctly when an error occurs during `EvalSubquery` building `Executor` #11811
  - Fix the issue that the query result might be incorrect when the number of rows in the outer table is greater than that in a single batch in Index Lookup Join; expand the functional scope of Index Lookup Join; `UnionScan` can be used as a subnode of `IndexJoin` #11843
  - Add the display of invalid keys (like `invalid encoded key flag 252`) in the `SHOW STAT_BUCKETS` syntax, for the situation where invalid keys might occur during the statistics feedback process #12098
- SQL Execution Engine
  - Fix some incorrect results (like `select cast(13835058000000000000 as double)`) caused by the number value that is first converted to `UINT` when the `CAST` function is converting the number value type #11712
  - Fix the issue that the calculation result might be incorrect when the dividend of the `DIV` calculation is a decimal and this calculation contains a negative number #11812
  - Add the `ConvertStrToIntStrict` function to fix the MySQL incompatibility issue caused by some strings being converted to the `INT` type when executing the `SELECT`/`EXPLAIN` statement #11892
  - Fix the issue that the `Explain` result might be incorrect caused by wrong configuration of `stmtCtx` when `EXPLAIN ... FOR CONNECTION` is used #11978
  - Fix the issue that the result returned by the `unaryMinus` function is incompatible with MySQL, caused by the non-decimal result when the integer result overflows #11990
  - Fix the issue that `last_insert_id()` might be incorrect, caused by the counting order when the `LOAD DATA` statement is being executed #11994
  - Fix the issue that `last_insert_id()` might be incorrect when the user writes auto-increment column data in an explicit-implicit mixed way #12001
  - Fix an over-quoted bug for the `JSON_UNQUOTE` function: only values enclosed by double quote marks (`"`) should be unquoted. For example, the result of "`SELECT JSON_UNQUOTE("\\\\")`" should be "`\\`" (not changed) #12096
- Server

  - Change `start ts` recorded in slow query logs from the last retry time to the first execution time when retrying TiDB transactions #11878
  - Add the number of keys of a transaction in `LockResolver` to avoid the scan operation on the whole Region and reduce costs of resolving locking when the number of keys is reduced #11889
  - Fix the issue that the `succ` field value might be incorrect in slow query logs #11886
  - Replace the `Index_ids` filed in slow query logs with the `Index_names` field to improve the usability of slow query logs #12063
  - Fix the connection break issue caused by TiDB parsing `-` into EOF Error when `Duration` contains `-` (like `select time( '--' )`) #11910
  - Remove an invalid Region from `RegionCache` more quickly to reduce the number of requests sent to this Region #11931
  - Fix the connection break issue caused by incorrectly handling the OOM panic issue when `oom-action = "cancel"` and OOM occurs in the `Insert Into ⋯ Select` syntax #12126

- DDL

  - Add the reverse scan interface for `tikvSnapshot` to efficiently query DDL history jobs. After using this interface, the execution time of `ADMIN SHOW DDL JOBS` is remarkably decreased #11789
  - Improve the `CREATE TABLE ... PRE_SPLIT_REGION` syntax: change the number of pre-splitting Regions from 2^(N-1) to 2^N when `PRE_SPLIT_REGION = N` #11797
  - Decrease the default parameter value for the background worker thread of the `Add Index` operation to avoid great impacts on online workloads #11875
  - Improve the `SPLIT TABLE` syntax behavior: generate N data Region(s) and one index Region when `SPLIT TABLE ... REGIONS N` is used to divide Regions #11929
  - Add the `split-region-max-num` parameter (`10000` by default) in the configuration file to make the maximum number of Regions allowed by the `SPLIT TABLE` syntax adjustable #12080
  - Fix the issue that the `CREATE TABLE` clause cannot be parsed by the downstream MySQL, caused by uncommented `PRE_SPLIT_REGIONS` in this clause when the system writes the binlog #12121
  - Add the `WHERE` sub-clause in `SHOW TABLE ⋯ REGIONS` and `SHOW TABLE .. INDEX ⋯ REGIONS` #12124

- Monitor

  - Add the `connection_transient_failure_count` monitoring metric to count gRPC connection errors of `tikvclient` #12092

9.3.3.2 TiKV

- Fix the incorrect result of counting keys in a Region in some cases #5415

- Add the `config-check` option in TiKV to check whether the TiKV configuration item is valid #5391
- Optimize the starting process to reduce jitters caused by restarting nodes #5277
- Optimize the resolving locking process in some cases to speed up resolving locking for transactions #5339
- Optimize the `get_txn_commit_info` process to speed up committing transactions #5062
- Simplify Raft-related logs #5425
- Resolve the issue that TiKV exits abnormally in some cases #5441

9.3.3.3 PD

- Add the `config-check` option in PD to check whether the PD configuration item is valid #1725
- Add the `remove-tombstone` command in pd-ctl to support clearing tombstone store records #1705
- Support actively pushing operators to speed up scheduling #1686

9.3.3.4 Tools

- TiDB Binlog
  - Add `worker-count` and `txn-batch` configuration items in Reparo to control the recovery speed #746
  - Optimize the memory usage of Drainer to improve the parallel execution efficiency #735
  - Fix the bug that Pump cannot quit normally in some cases #739
  - Optimize the processing logic of `LevelDB` in Pump to improve the execution efficiency of GC #720
- TiDB Lightning
  - Fix the bug that tidb-lightning might crash caused by re-importing data from the checkpoint #239

9.3.3.5 TiDB Ansible

- Update the Spark version to 2.4.3, and update the TiSpark version to 2.2.0 that is compatible with Spark 2.4.3 #914, #919
- Fix the issue that there is a long waiting time when the remote machine password has expired #937, #948

9.3.4 TiDB 2.1.16 Release Notes

Release date: August 15, 2019

TiDB version: 2.1.16

TiDB Ansible version: 2.1.16

9.3.4.1 TiDB

- SQL Optimizer
  - Fix the issue that row count is estimated inaccurately for the equal condition on the time column #11526
  - Fix the issue that `TIDB_INLJ` Hint does not take effect or take effect on the specified table #11361
  - Change the implementation of `NOT EXISTS` in a query from OUTER JOIN to ANTI JOIN to find a more optimized execution plan #11291
  - Support subqueries within `SHOW` statements, allowing syntaxes such as `SHOW COLUMNS FROM tbl WHERE FIELDS IN (SELECT 'a')` #11461
  - Fix the issue that the `SELECT ⋯ CASE WHEN ⋯ ELSE NULL ...` query gets an incorrect result caused by the constant folding optimization #11441
- SQL Execution Engine
  - Fix the issue that the `DATE_ADD` function gets a wrong result when `INTERVAL` is negative #11616
  - Fix the issue that the `DATE_ADD` function might get an incorrect result because it performs type conversion wrongly when it accepts an argument of the `FLOAT`, `DOUBLE`, or `DECIMAL` type #11628
  - Fix the issue that the error message is inaccurate when CAST(JSON AS SIGNED) overflows #11562
  - Fix the issue that other child nodes are not closed when one child node fails to be closed and returns an error during the process of closing Executor #11598
  - Support `SPLIT TABLE` statements that return the number of Regions that are successfully split and a finished percentage rather than an error when the scheduling is not finished for Region scatter before the timeout #11487
  - Make `REGEXP BINARY` function case sensitive to be compatible with MySQL #11505
  - Fix the issue that `NULL` is not returned correctly because the value of `YEAR` in the `DATE_ADD`/`DATE_SUB` result overflows when it is smaller than 0 or larger than 65535 #11477
  - Add in the slow query table a `Succ` field that indicates whether the execution succeeds #11412
  - Fix the MySQL incompatibility issue caused by fetching the current timestamp multiple times when a SQL statement involves calculations of the current time (such as `CURRENT_TIMESTAMP` or `NOW`) #11392

  - Fix the issue that the AUTO_INCREMENT columns do not handle the FLOAT or DOUBLE type #11389
  - Fix the issue that `NULL` is not returned correctly when the `CONVERT_TZ` function accepts an invalid argument #11357
  - Fix the issue that an error is reported by the `PARTITION BY LIST` statement. (Currently only the syntax is supported; when TiDB executes the statement, a regular table is created and a prompting message is provided) #11236
  - Fix the issue that `Mod(%)`, `Multiple(*)`, and `Minus(-)` operations return an inconsistent `0` result with that in MySQL when there are many decimal digits (such as `select 0.000 % 0.11234500000000000000`) #11353

- Server
  - Fix the issue that the plugin gets a `NULL` domain when `OnInit` is called back #11426
  - Fix the issue that the table information in a schema can still be obtained through the HTTP interface after the schema has been deleted #11586

- DDL
  - Disallow dropping indexes on auto-increment columns to avoid incorrect results of the auto-increment columns caused by this operation #11402
  - Fix the issue that the character set of the column is not correct when creating and modifying the table with different character sets and collations #11423
  - Fix the issue that the column schema might get wrong when `alter table ... ↪ set default...` and another DDL statement that modifies this column are executed in parallel #11374
  - Fix the issue that data fails to be backfilled when Generated Column A depends on Generated Column B and A is used to create an index #11538
  - Speed up `ADMIN CHECK TABLE` operations #11538

9.3.4.2 TiKV

- Support returning an error message when the client accesses a TiKV Region that is being closed #4820
- Support reverse `raw_scan` and `raw_batch_scan` interfaces #5148

9.3.4.3 Tools

- TiDB Binlog
  - Add the `ignore-txn-commit-ts` configuration item in Drainer to skip executing some statements in a transaction #697
  - Add the configuration item check on startup, which stops Pump and Drainer from running and returns an error message when meeting invalid configuration items #708

  - Add the `node-id` configuration in Drainer to specify Drainer's node ID #706

- TiDB Lightning
  - Fix the issue that `tikv_gc_life_time` fails to be changed back to its original value when 2 checksums are running at the same time #224

9.3.4.4 TiDB Ansible

- Add the `log4j` configuration file in Spark #842
- Update the tispark jar package to v2.1.2 #863
- Fix the issue that the Prometheus configuration file is generated in the wrong format when TiDB Binlog uses Kafka or ZooKeeper #845
- Fix the bug that PD fails to switch the Leader when executing the `rolling_update.yml` operation #888
- Optimize the logic of rolling updating PD nodes - upgrade Followers first and then the Leader - to improve stability #895

9.3.5 TiDB 2.1.15 Release Notes

Release date: July 16, 2019

TiDB version: 2.1.15

TiDB Ansible version: 2.1.15

9.3.5.1 TiDB

- Fix the issue that the `DATE_ADD` function returns wrong results due to incorrect alignment when dealing with microseconds #11289
- Fix the issue that an error is reported when the empty value in the string column is compared with `FLOAT` or `INT` #11279
- Fix the issue that the `INSERT` function fails to correctly return the `NULL` value when a parameter is `NULL` #11249
- Fix the issue that an error occurs when indexing the column of the non-string type and `0` length #11215
- Add the `SHOW TABLE REGIONS` statement to query the Region distribution of a table through SQL statements #11238
- Fix the issue that an error is reported when using the `UPDATE ⋯ SELECT` statement because the projection elimination is used to optimize rules in the `SELECT` subqueries #11254
- Add the `ADMIN PLUGINS ENABLE/ADMIN PLUGINS DISABLE` SQL statement to dynamically enable or disable plugins #11189
- Add the session connection information in the Audit plugin #11189
- Fix the panic issue that happens when a column is queried on multiple times and the returned result is `NULL` during point queries #11227

- Add the `tidb_scatter_region` configuration item to scatter table Regions when creating a table #11213
- Fix the data race issue caused by non-thread safe `rand.Rand` when using the `RAND` function #11170
- Fix the issue that the comparison result of integers and non-integers is incorrect in some cases #11191
- Support modifying the collation of a database or a table, but the character set of the database/table has to be UTF-8 or utf8mb4 #11085
- Fix the issue that the precision shown by the `SHOW CREATE TABLE` statement is incomplete when `CURRENT_TIMESTAMP` is used as the default value of the column and the float precision is specified #11087

9.3.5.2 TiKV

- Unify the log format #5083
- Improve the accuracy of Region's approximate size or keys in extreme cases to improve the accuracy of scheduling #5085

9.3.5.3 PD

- Unify the log format #1625

9.3.5.4 Tools

TiDB Binlog

- Optimize the Pump GC strategy and remove the restriction that the unconsumed binlog cannot be cleaned to make sure that the resources are not occupied for a long time #663

TiDB Lightning

- Fix the import error that happens when the column names specified by the SQL dump are not in lowercase #210

9.3.5.5 TiDB Ansible

- Add the `parse duration` and `compile duration` monitoring items in TiDB Dashboard to monitor the time that it takes to parse SQL statements and execute compilation #815

9.3.6 TiDB 2.1.14 Release Notes

Release date: July 04, 2019

TiDB version: 2.1.14

TiDB Ansible version: 2.1.14

9.3.6.1 TiDB

- Fix wrong query results caused by column pruning in some cases #11019
- Fix the wrongly displayed information in `db` and `info` columns of `show processlist` #11000
- Fix the issue that `MAX_EXECUTION_TIME` as a SQL hint and global variable does not work in some cases #10999
- Support automatically adjust the incremental gap allocated by auto-increment ID based on the load #10997
- Fix the issue that the `Distsql` memory information of `MemTracker` is not correctly cleaned when a query ends #10971
- Add the `MEM` column in the `information_schema.processlist` table to describe the memory usage of a query #10896
- Add the `max_execution_time` global system variable to control the maximum execution time of a query #10940
- Fix the panic caused by using unsupported aggregate functions #10911
- Add an automatic rollback feature for the last transaction when the `load data` statement fails #10862
- Fix the issue that TiDB returns a wrong result in some cases when the `OOMAction` configuration item is set to `Cancel` #11016
- Disable the `TRACE` statement to avoid the TiDB panic issue #11039
- Add the `mysql.expr_pushdown_blacklist` system table that dynamically enables/disables pushing down specific functions to Coprocessor #10998
- Fix the issue that the `ANY_VALUE` function does not work in the `ONLY_FULL_GROUP_BY` mode #10994
- Fix the incorrect evaluation caused by not doing a deep copy when evaluating the user variable of the string type #11043

9.3.6.2 TiKV

- Optimize processing the empty callback when processing the Raftstore message to avoid sending unnecessary message #4682

9.3.6.3 PD

- Adjust the log output level from `Error` to `Warning` when reading an invalid configuration item #1577

9.3.6.4 Tools

TiDB Binlog

- Reparo
  - Add the `safe-mode` configuration item, and support importing duplicated data after this item is enabled #662
- Pump
  - Add the `stop-write-at-available-space` configuration item to limit the available binlog space #659
  - Fix the issue that Garbage Collector does not work sometimes when the number of LevelDB L0 files is 0 #648
  - Optimize the algorithm of deleting log files to speed up releasing the space #648
- Drainer
  - Fix the failure to update `BIT` columns in the downstream #655

9.3.6.5 TiDB Ansible

- Add the precheck feature for the `ansible` command and its `jmespath` and `jinja2` dependency packages #807
- Add the `stop-write-at-available-space` parameter (10 GiB by default) in Pump, and stop writing binlog files in Pump when the available disk space is less than the parameter value #807

9.3.7 TiDB 2.1.13 Release Notes

Release date: June 21, 2019

TiDB version: 2.1.13

TiDB Ansible version: 2.1.13

9.3.7.1 TiDB

- Add a feature to use `SHARD_ROW_ID_BITS` to scatter row IDs when the column contains an `AUTO_INCREMENT` attribute to relieve the hotspot issue #10788
- Optimize the lifetime of invalid DDL metadata to speed up recovering the normal execution of DDL operations after upgrading the TiDB cluster #10789
- Fix the OOM issue in high concurrent scenarios caused by the failure to quickly release Coprocessor resources, resulted from the `execdetails.ExecDetails` pointer #10833
- Add the `update-stats` configuration item to control whether to update statistics #10772

- Add the following TiDB-specific syntax to support Region presplit to solve the hotspot issue:
- Add the `PRE_SPLIT_REGIONS` table option #10863
- Add the `SPLIT TABLE table_name INDEX index_name` syntax #10865
- Add the `SPLIT TABLE [table_name] BETWEEN (min_value...)AND (max_value ...)REGIONS [region_num]` syntax #10882
- Fix the panic issue caused by the `KILL` syntax in some cases #10879
- Improve the compatibility with MySQL for `ADD_DATE` in some cases #10718
- Fix the wrong estimation for the selectivity rate of the inner table selection in index join #10856

9.3.7.2 TiKV

- Fix the issue that incomplete snapshots are generated in the system caused by the iterator not checking the status #4940
- Add a feature to check the validity for the `block-size` configuration #4930

9.3.7.3 Tools

- TiDB Binlog
  - Fix the wrong offset issue caused by Pump not checking the returned value when it fails to write data #640
  - Add the `advertise-addr` configuration in Drainer to support the bridge mode in the container environment #634

9.3.8 TiDB 2.1.12 Release Notes

Release date: June 13, 2019

TiDB version: 2.1.12

TiDB Ansible version: 2.1.12

9.3.8.1 TiDB

- Fix the issue caused by unmatched data types when using the index query feedback #10755
- Fix the issue that the blob column is changed to the text column caused by charset altering in some cases #10745
- Fix the issue that the `GRANT` operation in the transaction mistakenly reports "Duplicate Entry" in some cases #10739
- Improve the compatibility with MySQL of the following features
  - The `DAYNAME` function #10732

  - The `MONTHNAME` function #10733
  - Support the 0 value for the `EXTRACT` function when processing `MONTH` #10702
  - The `DECIMAL` type can be converted to `TIMESTAMP` or `DATETIME` #10734

- Change the column charset while changing the table charset #10714
- Fix the overflow issue when converting a decimal to a float in some cases #10730
- Fix the issue that some extremely large messages report the "grpc: received message larger than max" error caused by inconsistent maximum sizes of messages sent/received by gRPC of TiDB and TiKV #10710
- Fix the panic issue caused by `ORDER BY` not filtering NULL in some cases #10488
- Fix the issue that values returned by the `UUID` function might be duplicate when multiple nodes exist #10711
- Change the value returned by `CAST(-num as datetime)` from `error` to NULL #10703
- Fix the issue that an unsigned histogram meets signed ranges in some cases #10695
- Fix the issue that an error is reported mistakenly for reading data when the statistics feedback meets the bigint unsigned primary key #10307
- Fix the issue that the result of `Show Create Table` for partitioned tables is not correctly displayed in some cases #10690
- Fix the issue that the calculation result of the `GROUP_CONCAT` aggregate function is not correct for some correlated subqueries #10670
- Fix the issue that the result is wrongly displayed when the memory table of slow queries parses the slow query log in some cases #10776

9.3.8.2 PD

- Fix the issue that etcd leader election is blocked in extreme conditions #1576

9.3.8.3 TiKV

- Fix the issue that Regions are not available during the leader transfer process in extreme conditions #4799
- Fix the issue that TiKV loses data when the power of the machine fails abnormally, caused by delayed data flush to the disk when receiving snapshots #4850

9.3.9 TiDB 2.1.11 Release Notes

Release date: June 03, 2019

TiDB version: 2.1.11

TiDB Ansible version: 2.1.11

9.3.9.1 TiDB

- Fix the issue that incorrect schema is used for `delete from join` #10595
- Fix the issue that the built-in `CONVERT()` may return incorrect field type #10263
- Merge non-overlapped feedback when updating bucket count #10569
- Fix calculation errors of `unix_timestamp()-unix_timestamp(now())` #10491
- Fix the incompatibility issue of `period_diff` with MySQL 8.0 #10501
- Skip `Virtual Column` when collecting statistics to avoid exceptions #10628
- Support the `SHOW OPEN TABLES` statement #10374
- Fix the issue that goroutine leak may happen in some cases #10656
- Fix the issue that setting the `tidb_snapshot` variable in some cases may cause incorrect parsing of time format #10637

9.3.9.2 PD

- Fix the issue that hots Region may fail to be scheduled due to `balance-region` #1551
- Set hotspot related scheduling priorities to high #1551
- Add two configuration items #1551
  - `hot-region-schedule-limit` to control the maximum number of concurrent hotspot scheduling tasks
  - `hot-region-cache-hits-threshold` to identify a hot Region

9.3.9.3 TiKV

- Fix the issue that the learner reads an empty index when there is only one leader and one learner #4751
- Process `ScanLock` and `ResolveLock` in the thread pool with a high priority to reduce their impacts on commands with a normal priority #4791
- Sync all files of received snapshots #4811

9.3.9.4 Tools

- TiDB Binlog
  - Limit data deletion speed during GC to avoid QPS degrading caused by `WritePause` #620

9.3.9.5 TiDB Ansible

- Add Drainer parameters #760

9.3.10 TiDB 2.1.10 Release Notes

Release date: May 22, 2019

TiDB version: 2.1.10

TiDB Ansible version: 2.1.10

9.3.10.1 TiDB

- Fix the issue that some abnormalities cause incorrect table schema when using `tidb_snapshot` to read the history data #10359
- Fix the issue that the `NOT` function causes wrong read results in some cases #10363
- Fix the wrong behavior of `Generated Column` in the `Replace` or `Insert on duplicate update` statement #10385
- Fix a bug of the `BETWEEN` function in the `DATE/DATETIME` comparison #10407
- Fix the issue that a single line of a slow log that is too long causes an error report when using the `SLOW_QUERY` table to query a slow log #10412
- Fix the issue that the result of `DATETIME` plus `INTERVAL` is not the same with that of MySQL in some cases #10416, #10418
- Add the check for the invalid time of February in a leap year #10417
- Execute the internal initialization operation limitation only in the DDL owner to avoid a large number of conflict error reports when initializing the cluster #10426
- Fix the issue that `DESC` is incompatible with MySQL when the default value of the output timestamp column is `default current_timestamp on update current_timestamp` #10337
- Fix the issue that an error occurs during the privilege check in the `Update` statement #10439
- Fix the issue that wrong calculation of `RANGE` causes a wrong result in the `CHAR` column in some cases #10455
- Fix the issue that the data might be overwritten after decreasing `SHARD_ROW_ID_BITS` #9868
- Fix the issue that `ORDER BY RAND()` does not return random numbers #10064
- Prohibit the `ALTER` statement modifying the precision of decimals #10458
- Fix the compatibility issue of the `TIME_FORMAT` function with MySQL #10474
- Check the parameter validity of `PERIOD_ADD` #10430
- Fix the issue that the behavior of the invalid `YEAR` string in TiDB is incompatible with that in MySQL #10493
- Support the `ALTER DATABASE` syntax #10503
- Fix the issue that the `SLOW_QUERY` memory engine reports an error when no `;` exists in the slow query statement #10536
- Fix the issue that the `Add index` operation in partitioned tables cannot be canceled in some cases #10533
- Fix the issue that the OOM panic cannot be recovered in some cases #10545
- Improve the security of the DDL operation rewriting the table metadata #10547

9.3.10.2 PD

- Fix the issue that the priority of the leader does not take effect #1533

9.3.10.3 TiKV

- Reject transferring the leader in a Region whose configuration has been changed recently to avoid transfer failure #4684
- Add the priority label for Coprocessor metrics #4643
- Fix the possible dirty read issue during transferring the leader #4724
- Fix the issue that `CommitMerge` causes the restart failure of TiKV in some cases #4615
- Fix unknown logs #4730

9.3.10.4 Tools

- TiDB Lightning
  - Add the retry feature when TiDB Lightning fails to send data to `importer` #176
- TiDB Binlog
  - Optimize the Pump storage log to facilitate troubleshooting #607

9.3.10.5 TiDB Ansible

- Update the configuration file of TiDB Lightning and add the `tidb_lightning_ctl` script #d3a4a368

9.3.11 TiDB 2.1.9 Release Notes

Release date: May 6, 2019

TiDB version: 2.1.9

TiDB-Ansible version: 2.1.9

9.3.11.1 TiDB

- Fix compatibility of the `MAKETIME` function when unsigned type overflows #10089
- Fix the stack overflow caused by constant folding in some cases #10189
- Fix the privilege check issue for `Update` when an alias exists in some cases #10157, #10326
- Track and control memory usage in DistSQL #10197
- Support specifying collation as `utf8mb4_0900_ai_ci` #10201

- Fix the wrong result issue of the `MAX` function when the primary key is of the Unsigned type #10209
- Fix the issue that NULL values can be inserted into NOT NULL columns in the non-strict SQL mode #10254
- Fix the wrong result issue of the `COUNT` function when multiple columns exist in `DISTINCT` #10270
- Fix the panic issue occurred when `LOAD DATA` parses irregular CSV files #10269
- Ignore the overflow error when the outer and inner join key types are inconsistent in `Index Lookup Join` #10244
- Fix the issue that a statement is wrongly judged as point-get in some cases #10299
- Fix the wrong result issue when the time type does not convert the time zone in some cases #10345
- Fix the issue that TiDB character set cases are inconsistent in some cases #10354
- Support controlling the number of rows returned by operator #9166
  - Selection & Projection #10110
  - `StreamAgg` & `HashAgg` #10133
  - `TableReader` & `IndexReader` & `IndexLookup` #10169
- Improve the slow query log
  - Add `SQL Digest` to distinguish similar SQL #10093
  - Add version information of statistics used by slow query statements #10220
  - Show memory consumption of a statement in slow query log #10246
  - Adjust the output format of Coprocessor related information so it can be parsed by pt-query-digest #10300
  - Fix the `#` character issue in slow query statements #10275
  - Add some information columns to the memory table of slow query statements #10317
  - Add the transaction commit time to slow query log #10310
  - Fix the issue some time formats cannot be parsed by pt-query-digest #10323

9.3.11.2 PD

- Support the GetOperator service #1514

9.3.11.3 TiKV

- Fix potential quorum changes when transferring leader #4604

9.3.11.4 Tools

- TiDB Binlog

  - Fix the issue that data replication is interrupted because data in the unsigned int type of primary key column are minus numbers #574
  - Remove the compression option when the downstream is `pb` and change the downstream name from `pb` to `file` #597
  - Fix the bug that Reparo introduced in 2.1.7 generates wrong `UPDATE` statements #576

- TiDB Lightning

  - Fix the bug that the bit type of column data is incorrectly parsed by the parser #164
  - Fill the lacking column data in dump files using row id or the default column value #174
  - Fix the Importer bug that some SST files fail to be imported but it still returns successful import result #4566
  - Support setting a speed limit in Importer when uploading SST files to TiKV #4607
  - Change Importer RocksDB SST compression method to `lz4` to reduce CPU consumption #4624

- sync-diff-inspector

  - Support checkpoint #227

9.3.11.5 TiDB-Ansible

- Update links in tidb-ansible documentation according to docs refactoring #740, #741
- Remove the `enable_slow_query_log` parameter in the `inventory.ini` file and output the slow query log to a separate log file by default #742

9.3.12 TiDB 2.1.8 Release Notes

Release date: April 12, 2019

TiDB version: 2.1.8

TiDB Ansible version: 2.1.8

9.3.12.1 TiDB

- Fix the issue that the processing logic of `GROUP_CONCAT` function is incompatible with MySQL when there is a NULL-valued parameter #9930
- Fix the equality check issue of decimal values in the `Distinct` mode #9931
- Fix the collation compatibility issue of the date, datetime, and timestamp types for the `SHOW FULL COLUMNS` statement

  - #9938

  - #10114

- Fix the issue that the row count estimation is inaccurate when the filtering condition contains correlated columns #9937
- Fix the compatibility issue between the `DATE_ADD` and `DATE_SUB` functions
  - #9963
  - #9966
- Support the `%H` format for the `STR_TO_DATE` function to improve compatibility #9964
- Fix the issue that the result is wrong when the `GROUP_CONCAT` function groups by a unique index #9969
- Return a warning when the Optimizer Hints contains an unmatched table name #9970
- Unify the log format to facilitate collecting logs using tools for analysis Unified Log Format
- Fix the issue that a lot of NULL values cause inaccurate statistics estimation #9979
- Fix the issue that an error is reported when the default value of the TIMESTAMP type is the boundary value #9987
- Validate the value of `time_zone` #10000
- Support the `2019.01.01` time format #10001
- Fix the issue that the row count estimation is displayed incorrectly in the result returned by the `EXPLAIN` statement in some cases #10044
- Fix the issue that `KILL TIDB [session id]` cannot instantly stop the execution of a statement in some cases #9976
- Fix the predicate pushdown issue of constant filtering conditions in some cases #10049
- Fix the issue that a read-only statement is not processed correctly in some cases #10048

9.3.12.2 PD

- Fix the issue that `regionScatterer` might generate an invalid `OperatorStep` #1482
- Fix the issue that a hot store makes incorrect statistics of keys #1487
- Fix the too short timeout issue of the `MergeRegion` operator #1495
- Add elapsed time metrics of the PD server handling TSO requests #1502

9.3.12.3 TiKV

- Fix the issue of wrong statistics of the read traffic #4441
- Fix the raftstore performance issue when too many Regions exist #4484
- Do not ingest files when the number of level 0 SST files exceeds `level_zero_slowdown_writes_trigg`
  ↪ `/2` #4464

9.3.12.4 Tools

- Optimize the order of importing tables for Lightning to reduce the effects of large tables executing `Checksum` and `Analyze` on the cluster during the importing process and improve the success rate of `Checksum` and `Analyze` #156

- Improve the encoding SQL performance by 50% for Lightning by directly parsing the data source file content to `types.Datum` of TiDB to avoid additional parsing working of the KV encoder #145
- Add the `storage.sync-log` configuration item in TiDB Binlog Pump to support flushing disks of the local storage asynchronously in Pump #529
- Support traffic compression of communication between TiDB Binlog Pump and Drainer #530
- Add the `syncer.sql-mode` configuration item in TiDB Binlog Drainer to support using different `sql-mode`s to parse DDL queries #513
- Add the `syncer.ignore-table` configuration item in TiDB Binlog Drainer to support filtering tables not to be replicated #526

9.3.12.5 TiDB Ansible

- Modify the version limit for the operating system and only support CentOS 7.0 or later and Red Hat 7.0 or later #734
- Add the feature of checking whether `epollexclusive` is supported in every OS #728
- Add the version limit for rolling update to prohibit upgrading a version of 2.0.1 or earlier to a version of 2.1 or later #728

9.3.13 TiDB 2.1.7 Release Notes

Release Date: March 28, 2019

TiDB version: 2.1.7

TiDB Ansible version: 2.1.7

9.3.13.1 TiDB

- Fix the issue of longer startup time when upgrading the program caused by canceling DDL operations #9768
- Fix the issue that the `check-mb4-value-in-utf8` configuration item is in the wrong position in the `config.example.toml` file #9852
- Improve the compatibility of the `str_to_date` built-in function with MySQL #9817
- Fix the compatibility issue of the `last_day` built-in function #9750
- Add the `tidb_table_id` column for `infoschema.tables` to facilitate getting `table_id` ↪ by using SQL statements and add the `tidb_indexes` system table to manage the relationship between Table and Index #9862
- Add a check about the null definition of Table Partition #9663
- Change the privileges required by `Truncate Table` from `Delete` to `Drop` to make it consistent with MySQL #9876
- Support using subqueries in the `DO` statement #9877
- Fix the issue that the `default_week_format` variable does not take effect in the `week` function #9753

- Support the plugin framework #9880, #9888
- Support checking the enabling state of binlog by using the `log_bin` system variable #9634
- Support checking the Pump/Drainer status by using SQL statements #9896
- Fix the compatibility issue about checking mb4 character on utf8 when upgrading TiDB #9887
- Fix the panic issue when the aggregate function calculates JSON data in some cases #9927

9.3.13.2 PD

- Fix the issue that the transferring leader step cannot be created in the balance-region when the number of replicas is one #1462

9.3.13.3 Tools

- Support replicating generated columns by using binlog

9.3.13.4 TiDB Ansible

Change the default retention time of Prometheus monitoring data to 30d

9.3.14 TiDB 2.1.6 Release Notes

On March 15, 2019, TiDB 2.1.6 is released. The corresponding TiDB Ansible 2.1.6 is also released. Compared with TiDB 2.1.5, this release has greatly improved the stability, the SQL optimizer, statistics, and the execution engine.

9.3.14.1 TiDB

- SQL Optimizer/Executor
  - Optimize planner to select the outer table based on cost when both tables are specified in Hint of `TIDB_INLJ` #9615
  - Fix the issue that `IndexScan` cannot be selected correctly in some cases #9587
  - Fix incompatibility with MySQL of check in the `agg` function in subqueries #9551
  - Make `show stats_histograms` only output valid columns to avoid panics #9502
- Server
  - Support the `log_bin` variable to enable/disable Binlog #9634
  - Add a sanity check for transactions to avoid false transaction commit #9559
  - Fix the issue that setting variables may lead to panic #9539
- DDL

- Fix the issue that the `Create Table Like` statement causes panic in some cases #9652
- Enable the `AutoSync` feature of etcd clients to avoid connection issues between TiDB and etcd in some cases #9600

9.3.14.2 TiKV

- Fix the issue that a `protobuf` parsing failure would in some cases cause a `StoreNotMatch` error #4303

9.3.14.3 Tools

- Lightning
  - Change the default `region-split-size` of importer to 512 MiB #4369
  - Save the intermediate SST previously cached in memory to the local disk to reduce memory usage #4369
  - Limit the memory usage of RocksDB #4369
  - Fix the issue that Regions are scattered before scheduling is finished #4369
  - Separate importing of data and indexes for large tables to effectively reduce time consumption when importing in batches #132
  - Support CSV #111
  - Fix the error of import failure due to non-alphanumeric characters in schema names #9547

9.3.15 TiDB 2.1.5 Release Notes

On February 28, 2019, TiDB 2.1.5 is released. The corresponding TiDB Ansible 2.1.5 is also released. Compared with TiDB 2.1.4, this release has greatly improved the stability, the SQL optimizer, statistics, and the execution engine.

9.3.15.1 TiDB

- SQL Optimizer/Executor
  - Make `SHOW CREATE TABLE` do not print the column charset information when the charset information of a column is the same with that of a table, to improve the compatibility of `SHOW CREATE TABLE` with MySQL #9306
  - Fix the panic or the wrong result of the `Sort` operator in some cases by extracting `ScalarFunc` from `Sort` to a `Projection` operator for computing to simplify the computing logic of `Sort` #9319
  - Remove the sorting field with constant values in the `Sort` operator #9335, #9440
  - Fix the data overflow issue when inserting data into an unsigned integer column #9339

  - Set `cast_as_binary` to `NULL` when the length of the target binary exceeds `max_allowed_packet` #9349
  - Optimize the constant folding process of `IF` and `IFNULL` #9351
  - Optimize the index selection of TiDB using skyline pruning to improve the stability of simple queries #9356
  - Support computing the selectivity of the `DNF` expression #9405
  - Fix the wrong SQL query result of `!=ANY()` and `=ALL()` in some cases #9403
  - Fix the panic or the wrong result when the Join Key types of two tables on which the `Merge Join` operation is performed are different #9438
  - Fix the issue that the result of the `RAND()` function is not compatible with MySQL #9446
  - Refactor the logic of `Semi Join` processing `NULL` and the empty result set to get the correct result and improve the compatibility with MySQL #9449

- Server

  - Add the `tidb_constraint_check_in_place` system variable to check the data uniqueness constraint when executing the `INSERT` statement #9401
  - Fix the issue that the value of the `tidb_force_priority` system variable is different from that set in the configuration file #9347
  - Add the `current_db` field in general logs to print the name of the currently used database #9346
  - Add an HTTP API of obtaining the table information with the table ID #9408
  - Fix the issue that `LOAD DATA` loads incorrect data in some cases #9414
  - Fix the issue that it takes a long time to build a connection between the MySQL client and TiDB in some cases #9451

- DDL

  - Fix some issues when canceling the `DROP COLUMN` operation #9352
  - Fix some issues when canceling the `DROP` or `ADD` partitioned table operation #9376
  - Fix the issue that `ADMIN CHECK TABLE` mistakenly reports the data index inconsistency in some cases #9399
  - Fix the time zone issue of the `TIMESTAMP` default value #9108

9.3.15.2 PD

- Provide the `exclude_tombstone_stores` option in the `GetAllStores` interface to remove the Tombstone store from the returned result #1444

9.3.15.3 TiKV

- Fix the issue that Importer fails to import data in some cases #4223
- Fix the `KeyNotInRegion` error in some cases #4125
- Fix the panic issue caused by Region merge in some cases #4235
- Add the detailed `StoreNotMatch` error message #3885

9.3.15.4 Tools

- Lightning
  - Do not report an error or exit when a Tombstone store exists in the cluster #4223
- TiDB Binlog
  - Update the DDL binlog replication plan to guarantee the correctness of DDL event replication #9304

9.3.16 TiDB 2.1.4 Release Notes

On February 15, 2019, TiDB 2.1.4 is released. The corresponding TiDB Ansible 2.1.4 is also released. Compared with TiDB 2.1.3, this release has greatly improved the stability, the SQL optimizer, statistics, and the execution engine.

9.3.16.1 TiDB

- SQL Optimizer/Executor
  - Fix the issue that the `VALUES` function does not handle the FLOAT type correctly #9223
  - Fix the wrong result issue when casting Float to String in some cases #9227
  - Fix the wrong result issue of the `FORMAT` function in some cases #9235
  - Fix the panic issue when handling the Join query in some cases #9264
  - Fix the issue that the `VALUES` function does not handle the ENUM type correctly #9280
  - Fix the wrong result issue of `DATE_ADD/DATE_SUB` in some cases #9284
- Server
  - Optimize the "reload privilege success" log and change it to the DEBUG level #9274
- DDL
  - Change `tidb_ddl_reorg_worker_cnt` and `tidb_ddl_reorg_batch_size` to global variables #9134
  - Fix the bug caused by adding an index to a generated column in some abnormal conditions #9289

9.3.16.2 TiKV

- Fix the duplicate write issue when closing TiKV #4146
- Fix the abnormal result issue of the event listener in some cases #4132

9.3.16.3 Tools

- Lightning
  - Optimize the memory usage #107, #108
  - Remove the chunk separation of dump files to avoid an extra parsing of dump files #109
  - Limit the I/O concurrency of reading dump files, to avoid performance degradation caused by too many cache misses #110
  - Support importing data in batches for a single table, to improve import stability #110
  - Enable auto compactions in the import mode in TiKV #4199
  - Support disabling the TiKV periodic Level-1 compaction parameter, because the Level-1 compaction is automatically executed in the import mode when the TiKV cluster version is 2.1.4 or later #119
  - Limit the number of import engines to avoid consuming too much importer disk space #119
- Support splitting chunks using the TiDB statistics in sync-diff-inspector #197

9.3.17 TiDB 2.1.3 Release Notes

On January 28, 2019, TiDB 2.1.3 is released. The corresponding TiDB Ansible 2.1.3 is also released. Compared with TiDB 2.1.2, this release has great improvement in system stability, SQL optimizer, statistics information, and execution engine.

9.3.17.1 TiDB

- SQL Optimizer/Executor
  - Fix the panic issue of Prepared Plan Cache in some cases #8826
  - Fix the issue that Range computing is wrong when the index is a prefix index #8851
  - Make `CAST(str AS TIME(N))` return null if the string is in the illegal `TIME` format when `SQL_MODE` is not strict #8966
  - Fix the panic issue of Generated Column during the process of `UPDATE` in some cases #8980
  - Fix the upper bound overflow issue of the statistics histogram in some cases #8989
  - Support Range for `_tidb_rowid` construction queries, to avoid full table scan and reduce cluster stress #9059
  - Return an error when the `CAST(AS TIME)` precision is too big #9058
  - Allow using `Sort Merge Join` in the Cartesian product #9037
  - Fix the issue that the statistics worker cannot resume after the panic in some cases #9085
  - Fix the issue that `Sort Merge Join` returns the wrong result in some cases #9046

  - Support returning the JSON type in the `CASE` clause #8355

- Server
  - Return a warning instead of an error when the non-TiDB hint exists in the comment #8766
  - Verify the validity of the configured TIMEZONE value #8879
  - Optimize the `QueryDurationHistogram` metrics item to display more statement types #8875
  - Fix the lower bound overflow issue of bigint in some cases #8544
  - Support the `ALLOW_INVALID_DATES` SQL mode #9110

- DDL
  - Fix a `RENAME TABLE` compatibility issue to keep the behavior consistent with that of MySQL #8808
  - Support making concurrent changes of `ADD INDEX` take effect immediately #8786
  - Fix the `UPDATE` panic issue during the process of `ADD COLUMN` in some cases #8906
  - Fix the issue of concurrently creating Table Partition in some cases #8902
  - Support converting the `utf8` character set to `utf8mb4` #8951 #9152
  - Fix the issue of Shard Bits overflow #8976
  - Support outputting the column character sets in `SHOW CREATE TABLE` #9053
  - Fix the issue of the maximum length limit of the varchar type column in `utf8mb4` #8818
  - Support `ALTER TABLE TRUNCATE TABLE PARTITION` #9093
  - Resolve the charset when the charset is not provided #9147

9.3.17.2 PD

- Fix the Watch issue related to leader election #1396

9.3.17.3 TiKV

- Support obtaining the monitoring information using the HTTP method #3855
- Fix the NULL issue of `data_format` #4075
- Add verifying the range for scan requests #4124

9.3.17.4 Tools

- TiDB Binlog
  - Fix the `no available pump` issue while TiDB is started or restarted #157
  - Enable outputting the Pump client log #165
  - Fix the data inconsistency issue caused by the unique key containing the NULL value when the table only has the unique key and does not have the primary key

9.3.18 TiDB 2.1.2 Release Notes

On December 22, 2018, TiDB 2.1.2 is released. The corresponding TiDB Ansible 2.1.2 is also released. Compared with TiDB 2.1.1, this release has great improvement in system compatibility and stability.

9.3.18.1 TiDB

- Make TiDB compatible with TiDB Binlog of the Kafka version #8747
- Improve the exit mechanism of TiDB in a rolling update #8707
- Fix the panic issue caused by adding the index for the generated column in some cases #8676
- Fix the issue that the optimizer cannot find the optimal query plan when `TIDB_SMJ` ↪ `Hint` exists in the SQL statement in some cases #8729
- Fix the issue that `AntiSemiJoin` returns an incorrect result in some cases #8730
- Improve the valid character check of the `utf8` character set #8754
- Fix the issue that the field of the time type might return an incorrect result when the write operation is performed before the read operation in a transaction #8746

9.3.18.2 PD

- Fix the Region information update issue about Region merge #1377

9.3.18.3 TiKV

- Support the configuration format in the unit of `DAY` (`d`) and fix the configuration compatibility issue #3931
- Fix the possible panic issue caused by `Approximate Size Split` #3942
- Fix two issues about Region merge #3822, #3873

9.3.18.4 Tools

- TiDB Lightning
  - Make TiDB 2.1.0 the minimum cluster version supported by Lightning
  - Fix the content error of the file involving parsed `JSON` data in Lightning #144
  - Fix the issue that `Too many open engines` occurs after the checkpoint is used to restart Lightning
- TiDB Binlog
  - Eliminate some bottlenecks of Drainer writing data to Kafka
  - Support the Kafka version of TiDB Binlog

9.3.19 TiDB 2.1.1 Release Notes

On December 12, 2018, TiDB 2.1.1 is released. Compared with TiDB 2.1.0, this release has great improvement in stability, SQL optimizer, statistics information, and execution engine.

9.3.19.1 TiDB

- SQL Optimizer/Executor
  - Fix the round error of the negative date #8574
  - Fix the issue that the `uncompress` function does not check the data length #8606
  - Reset bind arguments of the `prepare` statement after the `execute` command is executed #8652
  - Support automatically collecting the statistics information of a partition table #8649
  - Fix the wrongly configured integer type when pushing down the `abs` function #8628
  - Fix the data race on the JSON column #8660
- Server
  - Fix the issue that the transaction obtained TSO is incorrect when PD breaks down #8567
  - Fix the bootstrap failure caused by the statement that does not conform to ANSI standards #8576
  - Fix the issue that incorrect parameters are used in transaction retries #8638
- DDL
  - Change the default character set and collation of tables into `utf8mb4` #8590
  - Add the `ddl_reorg_batch_size` variable to control the speed of adding indexes #8614
  - Make the character set and collation options content in DDL case-insensitive #8611
  - Fix the issue of adding indexes for generated columns #8655

9.3.19.2 PD

- Fix the issue that some configuration items cannot be set to `0` in the configuration file #1334
- Check the undefined configuration when starting PD #1362
- Avoid transferring the leader to a newly created peer, to optimize the possible delay #1339
- Fix the issue that `RaftCluster` cannot stop caused by deadlock #1370

9.3.19.3 TiKV

- Avoid transferring the leader to a newly created peer, to optimize the possible delay #3878

9.3.19.4 Tools

- Lightning
  - Optimize the `analyze` mechanism on imported tables to increase the import speed
  - Support storing the checkpoint information to a local file
- TiDB Binlog
  - Fix the output bug of pb files that a table only with the primary key column cannot generate the pb event

9.3.20 TiDB 2.1 GA Release Notes

On November 30, 2018, TiDB 2.1 GA is released. See the following updates in this release. Compared with TiDB 2.0, this release has great improvements in stability, performance, compatibility, and usability.

9.3.20.1 TiDB

- SQL Optimizer
  - Optimize the selection range of `Index Join` to improve the execution performance
  - Optimize the selection of outer table for `Index Join` and use the table with smaller estimated value of Row Count the as the outer table
  - Optimize Join Hint `TIDB_SMJ` so that Merge Join can be used even without proper index available
  - Optimize Join Hint `TIDB_INLJ` to specify the Inner table to Join
  - Optimize correlated subquery, push down Filter, and extend the index selection range, to improve the efficiency of some queries by orders of magnitude
  - Support using Index Hint and Join Hint in the `UPDATE` and `DELETE` statement
  - Support pushing down more functions: `ABS/CEIL/FLOOR/IS TRUE/IS FALSE`
  - Optimize the constant folding algorithm for the `IF` and `IFNULL` built-in functions
  - Optimize the output of the `EXPLAIN` statement and use hierarchy structure to show the relationship between operators
- SQL executor

  - Refactor all the aggregation functions and improve execution efficiency of the `Stream` and `Hash` aggregation operators
  - Implement the parallel `Hash Aggregate` operators and improve the computing performance by 350% in some scenarios
  - Implement the parallel `Project` operators and improve the performance by 74% in some scenarios
  - Read the data of the Inner table and Outer table of `Hash Join` concurrently to improve the execution performance
  - Optimize the execution speed of the `REPLACE INTO` statement and increase the performance nearly by 10 times
  - Optimize the memory usage of the time data type and decrease the memory usage of the time data type by fifty percent
  - Optimize the point select performance and improve the point select efficiency result of Sysbench by 60%
  - Improve the performance of TiDB on inserting or updating wide tables by 20 times
  - Support configuring the memory upper limit of a single statement in the configuration file
  - Optimize the execution of Hash Join, if the Join type is Inner Join or Semi Join and the inner table is empty, return the result without reading data from the outer table
  - Support using the `EXPLAIN ANALYZE` statement to check the runtime statistics including the execution time and the number of returned rows of each operator

- Statistics

  - Support enabling auto ANALYZE statistics only during certain period of the day
  - Support updating the table statistics automatically according to the feedback of the queries
  - Support configuring the number of buckets in the histogram using the `ANALYZE TABLE WITH BUCKETS` statement
  - Optimize the Row Count estimation algorithm using histogram for mixed queries of equality query and range queries

- Expressions

  - Support following built-in function:
    * `json_contains`
    * `json_contains_path`
    * `encode/decode`

- Server
  - Support queuing the locally conflicted transactions within tidb-server instance to optimize the performance of conflicted transactions
  - Support Server Side Cursor
  - Add the HTTP API
    * Scatter the distribution of table Regions in the TiKV cluster
    * Control whether to open the `general log`
    * Support modifying the log level online
    * Check the TiDB cluster information
  - Add the `auto_analyze_ratio` system variables to contorl the ratio of Analyze
  - Add the `tidb_retry_limit` system variable to control the automatic retry times of transactions
  - Add the `tidb_disable_txn_auto_retry` system variable to control whether the transaction retries automatically
  - Support using`admin show slow` statement to obtain the slow queries
  - Add the `tidb_slow_log_threshold` environment variable to set the threshold of slow log automatically
  - Add the `tidb_query_log_max_len` environment variable to set the length of the SQL statement to be truncated in the log dynamically
- DDL
  - Support the parallel execution of the Add index statement and other statements to avoid the time consuming Add index operation blocking other operations
  - Optimize the execution speed of `ADD INDEX` and improve it greatly in some scenarios
  - Support the `select tidb_is_ddl_owner()` statement to facilitate deciding whether TiDB is `DDL Owner`
  - Support the `ALTER TABLE FORCE` syntax
  - Support the `ALTER TABLE RENAME KEY TO` syntax
  - Add the table name and database name in the output information of `admin show ddl jobs`
  - Support using the `ddl/owner/resign` HTTP interface to release the DDL owner and start electing a new DDL owner
- Compatibility
  - Support more MySQL syntaxes
  - Make the `BIT` aggregate function support the `ALL` parameter

- Support the `SHOW PRIVILEGES` statement
- Support the `CHARACTER SET` syntax in the `LOAD DATA` statement
- Support the `IDENTIFIED WITH` syntax in the `CREATE USER` statement
- Support the `LOAD DATA IGNORE LINES` statement
- The `Show ProcessList` statement returns more accurate information

9.3.20.2 Placement Driver (PD)

- Optimize availability
  - Introduce the version control mechanism and support rolling update of the cluster compatibly
  - Enable `Raft PreVote` among PD nodes to avoid leader reelection when network recovers after network isolation
  - Enable `raft learner` by default to lower the risk of unavailable data caused by machine failure during scheduling
  - TSO allocation is no longer affected by the system clock going backwards
  - Support the `Region merge` feature to reduce the overhead brought by metadata
- Optimize the scheduler
  - Optimize the processing of Down Store to speed up making up replicas
  - Optimize the hotspot scheduler to improve its adaptability when traffic statistics information jitters
  - Optimize the start of Coordinator to reduce the unnecessary scheduling caused by restarting PD
  - Optimize the issue that Balance Scheduler schedules small Regions frequently
  - Optimize Region merge to consider the number of rows within the Region
  - Add more commands to control the scheduling policy
  - Improve PD simulator to simulate the scheduling scenarios
- API and operation tools
  - Add the `GetPrevRegion` interface to support the `TiDB reverse scan` feature
  - Add the `BatchSplitRegion` interface to speed up TiKV Region splitting
  - Add the `GCSafePoint` interface to support distributed GC in TiDB
  - Add the `GetAllStores` interface, to support distributed GC in TiDB
  - pd-ctl supports:
    * using statistics for Region split

    * calling `jq` to format the JSON output
    * checking the Region information of the specified store
    * checking topN Region list sorted by versions
    * checking topN Region list sorted by size
    * more precise TSO encoding
  - pd-recover doesn't need to provide the `max-replica` parameter

- Metrics
  - Add related metrics for `Filter`
  - Add metrics about etcd Raft state machine

- Performance
  - Optimize the performance of Region heartbeat to reduce the memory overhead brought by heartbeats
  - Optimize the Region tree performance
  - Optimize the performance of computing hotspot statistics

9.3.20.3 TiKV

- Coprocessor
  - Add more built-in functions
  - Add Coprocessor `ReadPool` to improve the concurrency in processing the requests
  - Fix the time function parsing issue and the time zone related issues
  - Optimize the memory usage for pushdown aggregation computing

- Transaction
  - Optimize the read logic and memory usage of MVCC to improve the performance of the scan operation and the performance of full table scan is 1 time better than that in TiDB 2.0
  - Fold the continuous Rollback records to ensure the read performance
  - Add the `UnsafeDestroyRange` API to support to collecting space for the dropping table/index
  - Separate the GC module to reduce the impact on write
  - Add the `upper bound` support in the `kv_scan` command

- Raftstore

  - Improve the snapshot writing process to avoid RocksDB stall
  - Add the `LocalReader` thread to process read requests and reduce the delay for read requests
  - Support `BatchSplit` to avoid large Region brought by large amounts of write
  - Support `Region Split` according to statistics to reduce the I/O overhead
  - Support `Region Split` according to the number of keys to improve the concurrency of index scan
  - Improve the Raft message process to avoid unnecessary delay brought by `Region Split`
  - Enable the `PreVote` feature by default to reduce the impact of network isolation on services

- Storage Engine

  - Fix the `CompactFiles`bug in RocksDB and reduce the impact on importing data using Lightning
  - Upgrade RocksDB to v5.15 to fix the possible issue of snapshot file corruption
  - Improve `IngestExternalFile` to avoid the issue that flush could block write

- tikv-ctl

  - Add the `ldb` command to diagnose RocksDB related issues
  - The `compact` command supports specifying whether to compact data in the bottommost level

9.3.20.4 Tools

- Fast full import of large amounts of data: TiDB Lightning
- Support new TiDB Binlog

9.3.20.5 Upgrade caveat

- TiDB 2.1 does not support downgrading to v2.0.x or earlier due to the adoption of the new storage engine
- Parallel DDL is enabled in TiDB 2.1, so the clusters with TiDB version earlier than 2.0.1 cannot upgrade to 2.1 using rolling update. You can choose either of the following two options:

  - Stop the cluster and upgrade to 2.1 directly
  - Roll update to 2.0.1 or later 2.0.x versions, and then roll update to the 2.1 version

- If you upgrade from TiDB 2.0.6 or earlier to TiDB 2.1, check if there is any ongoing DDL operation, especially the time consuming `Add Index` operation, because the DDL operations slow down the upgrading process. If there is ongoing DDL operation, wait for the DDL operation finishes and then roll update.

9.3.21 TiDB 2.1 RC5 Release Notes

On November 12, 2018, TiDB 2.1 RC5 is released. Compared with TiDB 2.1 RC4, this release has great improvement in stability, SQL optimizer, statistics information, and execution engine.

9.3.21.1 TiDB

- SQL Optimizer
  - Fix the issue that `IndexReader` reads the wrong handle in some cases #8132
  - Fix the issue occurred while the `IndexScan Prepared` statement uses `Plan Cache` #8055
  - Fix the issue that the result of the `Union` statement is unstable #8165
- SQL Execution Engine
  - Improve the performance of TiDB on inserting or updating wide tables #8024
  - Support the unsigned `int` flag in the `Truncate` built-in function #8068
  - Fix the error occurred while converting JSON data to the decimal type #8109
  - Fix the error occurred when you `Update` the float type #8170
- Statistics
  - Fix the incorrect statistics issue during point queries in some cases #8035
  - Fix the selectivity estimation of statistics for primary key in some cases #8149
  - Fix the issue that the statistics of deleted tables are not cleared up for a long period of time #8182
- Server
  - Improve the readability of logs and make logs better
    * #8063
    * #8053
    * #8224
  - Fix the error occurred when obtaining the table data of `infoschema.profiling` #8096
  - Replace the unix socket with the pumps client to write binlogs #8098
  - Add the threshold value for the `tidb_slow_log_threshold` environment variable, which dynamically sets the slow log #8094
  - Add the original length of a SQL statement truncated while the `tidb_query_log_max_len` ↪ environment variable dynamically sets logs #8200

  - Add the `tidb_opt_write_row_id` environment variable to control whether to allow writing `_tidb_rowid` #8218
  - Add an upper bound to the `Scan` command of ticlient, to avoid overbound scan #8081, #8247

- DDL
  - Fix the issue that executing DDL statements in transactions encounters an error in some cases #8056
  - Fix the issue that executing `truncate table` in partition tables does not take effect #8103
  - Fix the issue that the DDL operation does not roll back correctly after being cancelled in some cases #8057
  - Add the `admin show next_row_id` command to return the next available row ID #8268

9.3.21.2 PD

- Fix the issues related to `pd-ctl` reading the Region key
  - #1298
  - #1299
  - #1308
- Fix the issue that the `regions/check` API returns the wrong result #1311
- Fix the issue that PD cannot restart join after a PD join failure #1279
- Fix the issue that `watch leader` might lose events in some cases #1317

9.3.21.3 TiKV

- Improve the error message of `WriteConflict` #3750
- Add the panic mark file #3746
- Downgrade grpcio to avoid the segment fault issue caused by the new version of gRPC #3650
- Add an upper limit to the `kv_scan` interface #3749

9.3.21.4 Tools

- Support the TiDB-Binlog cluster, which is not compatible with the older version of binlog #8093, documentation

9.3.22 TiDB 2.1 RC4 Release Notes

On October 23, 2018, TiDB 2.1 RC4 is released. Compared with TiDB 2.1 RC3, this release has great improvement in stability, SQL optimizer, statistics information, and execution engine.

9.3.22.1 TiDB

- SQL Optimizer
  - Fix the issue that column pruning of `UnionAll` is incorrect in some cases #7941
  - Fix the issue that the result of the `UnionAll` operator is incorrect in some cases #8007
- SQL Execution Engine
  - Fix the precision issue of the `AVG` function #7874
  - Support using the `EXPLAIN ANALYZE` statement to check the runtime statistics including the execution time and the number of returned rows of each operator during the query execution process #7925
  - Fix the panic issue of the `PointGet` operator when a column of a table appears multiple times in the result set #7943
  - Fix the panic issue caused by too large values in the `Limit` subclause #8002
  - Fix the panic issue during the execution process of the `AddDate`/`SubDate` statement in some cases #8009
- Statistics
  - Fix the issue of judging the prefix of the histogram low-bound of the combined index as out of range #7856
  - Fix the memory leak issue caused by statistics collecting #7873
  - Fix the panic issue when the histogram is empty #7928
  - Fix the issue that the histogram bound is out of range when the statistics is being uploaded #7944
  - Limit the maximum length of values in the statistics sampling process #7982
- Server
  - Refactor Latch to avoid misjudgment of transaction conflicts and improve the execution performance of concurrent transactions #7711
  - Fix the panic issue caused by collecting slow queries in some cases #7874
  - Fix the panic issue when `ESCAPED BY` is an empty string in the `LOAD DATA` statement #8005
  - Complete the "coprocessor error" log information #8006
- Compatibility
  - Set the `Command` field of the `SHOW PROCESSLIST` result to `Sleep` when the query is empty #7839
- Expressions
  - Fix the constant folding issue of the `SYSDATE` function #7895
  - Fix the issue that `SUBSTRING_INDEX` panics in some cases #7897
- DDL

  - Fix the stack overflow issue caused by throwing the `invalid ddl job type` error #7958
  - Fix the issue that the result of `ADMIN CHECK TABLE` is incorrect in some cases #7975

9.3.22.2 PD

- Fix the issue that the tombstone TiKV is not removed from Grafana #1261
- Fix the data race issue when grpc-go configures the status #1265
- Fix the issue that the PD server gets stuck caused by etcd startup failure #1267
- Fix the issue that data race might occur during leader switching #1273
- Fix the issue that extra warning logs might be output when TiKV becomes tombstone #1280

9.3.22.3 TiKV

- Optimize the RocksDB Write stall issue caused by applying snapshots #3606
- Add raftstore `tick` metrics #3657
- Upgrade RocksDB and fix the Write block issue and that the source file might be damaged by the Write operation when performing `IngestExternalFile` #3661
- Upgrade grpcio and fix the issue that "too many pings" is wrongly reported #3650

9.3.23 TiDB 2.1 RC3 Release Notes

On September 29, 2018, TiDB 2.1 RC3 is released. Compared with TiDB 2.1 RC2, this release has great improvement in stability, compatibility, SQL optimizer, and execution engine.

9.3.23.1 TiDB

- SQL Optimizer
  - Fix the incorrect result issue when a statement contains embedded `LEFT OUTER JOIN` #7689
  - Enhance the optimization rule of predicate pushdown on the `JOIN` statement #7645
  - Fix the optimization rule of predicate pushdown for the `UnionScan` operator #7695
  - Fix the issue that the unique key property of the `Union` operator is not correctly set #7680
  - Enhance the optimization rule of constant folding #7696
  - Optimize the data source in which the filter is null after propagation to table dual #7756

- SQL Execution Engine
  - Optimize the performance of read requests in a transaction #7717
  - Optimize the cost of allocating Chunk memory in some executors #7540
  - Fix the "index out of range" panic caused by the columns where point queries get all NULL values #7790
- Server
  - Fix the issue that the memory quota in the configuration file does not take effect #7729
  - Add the `tidb_force_priority` system variable to set the execution priority for each statement #7694
  - Support using the `admin show slow` statement to obtain the slow query log #7785
- Compatibility
  - Fix the issue that the result of `charset/collation` is incorrect in `information_schema` ↪ `.schemata` #7751
  - Fix the issue that the value of the `hostname` system variable is empty #7750
- Expressions
  - Support the `init_vecter` argument in the `AES_ENCRYPT/AES_DECRYPT` built-in function #7425
  - Fix the issue that the result of `Format` is incorrect in some expressions #7770
  - Support the `JSON_LENGTH` built-in function #7739
  - Fix the incorrect result issue when casting the unsigned integer type to the decimal type #7792
- DML
  - Fix the issue that the result of the `INSERT ⋯ ON DUPLICATE KEY UPDATE` statement is incorrect while updating the unique key #7675
- DDL
  - Fix the issue that the index value is not converted between time zones when you create a new index on a new column of the timestamp type #7724
  - Support appending new values for the enum type #7767
  - Support creating an etcd session quickly, which improves the cluster availability after network isolation #7774

9.3.23.2 PD

- New feature
  - Add the API to get the Region list by size in reverse order #1254

- Improvement
  - Return more detailed information in the Region API #1252
- Bug fix
  - Fix the issue that `adjacent-region-scheduler` might lead to a crash after PD switches the leader #1250

9.3.23.3 TiKV

- Performance
  - Optimize the concurrency for coprocessor requests #3515
- New features
  - Add the support for Log functions #3603
  - Add the support for the `sha1` function #3612
  - Add the support for the `truncate_int` function #3532
  - Add the support for the `year` function #3622
  - Add the support for the `truncate_real` function #3633
- Bug fixes
  - Fix the reporting error behavior related to time functions #3487, #3615
  - Fix the issue that the time parsed from string is inconsistent with that in TiDB #3589

9.3.24 TiDB 2.1 RC2 Release Notes

On September 14, 2018, TiDB 2.1 RC2 is released. Compared with TiDB 2.1 RC1, this release has great improvement in stability, SQL optimizer, statistics information, and execution engine.

9.3.24.1 TiDB

- SQL Optimizer
  - Put forward a proposal of the next generation Planner #7543
  - Improve the optimization rules of constant propagation #7276
  - Enhance the computing logic of `Range` to enable it to handle multiple `IN` or `EQUAL` conditions simultaneously #7577
  - Fix the issue that the estimation result of `TableScan` is incorrect when `Range` is empty #7583
  - Support the `PointGet` operator for the `UPDATE` statement #7586
  - Fix the panic issue during the process of executing the `FirstRow` aggregate function in some conditions #7624

- SQL Execution Engine
  - Fix the potential `DataRace` issue when the `HashJoin` operator encounters an error #7554
  - Make the `HashJoin` operator read the inner table and build the hash table simultaneously #7544
  - Optimize the performance of Hash aggregate operators #7541
  - Optimize the performance of Join operators #7493, #7433
  - Fix the issue that the result of `UPDATE JOIN` is incorrect when the Join order is changed #7571
  - Improve the performance of Chunk's iterator #7585
- Statistics
  - Fix the issue that the auto Analyze work repeatedly analyzes the statistics #7550
  - Fix the statistics update error that occurs when there is no statistics change #7530
  - Use the RC isolation level and low priority when building `Analyze` requests #7496
  - Support enabling statistics auto-analyze on certain period of a day #7570
  - Fix the panic issue when logging the statistics information #7588
  - Support configuring the number of buckets in the histogram using the `ANALYZE TABLE WITH BUCKETS` statement #7619
  - Fix the panic issue when updating an empty histogram #7640
  - Update `information_schema.tables.data_length` using the statistics information #7657
- Server
  - Add Trace related dependencies #7532
  - Enable the `mutex profile` feature of Golang #7512
  - The `Admin` statement requires the `Super_priv` privilege #7486
  - Forbid users to `Drop` crucial system tables #7471
  - Switch from `juju/errors` to `pkg/errors` #7151
  - Complete the functional prototype of SQL Tracing #7016
  - Remove the goroutine pool #7564
  - Support viewing the goroutine information using the `USER1` signal #7587
  - Set the internal SQL to high priority while TiDB is started #7616
  - Use different labels to filter internal SQL and user SQL in monitoring metrics #7631
  - Store the top 30 slow queries in the last week to the TiDB server #7646
  - Put forward a proposal of setting the global system time zone for the TiDB cluster #7656
  - Enrich the error message of "GC life time is shorter than transaction duration" #7658
  - Set the global system time zone when starting the TiDB cluster #7638
- Compatibility

  - Add the unsigned flag for the `Year` type #7542
  - Fix the issue of configuring the result length of the `Year` type in the `Prepare` ↪ `/Execute` mode #7525
  - Fix the issue of inserting zero timestamp in the `Prepare/Execute` mode #7506
  - Fix the error handling issue of the integer division #7492
  - Fix the compatibility issue when processing `ComStmtSendLongData` #7485
  - Fix the error handling issue during the process of converting string to integer #7483
  - Optimize the accuracy of values in the `information_schema.columns_in_table` table #7463
  - Fix the compatibility issue when writing or updating the string type of data using the MariaDB client #7573
  - Fix the compatibility issue of aliases of the returned value #7600
  - Fix the issue that the `NUMERIC_SCALE` value of the float type is incorrect in the `information_schema.COLUMNS` table #7602
  - Fix the issue that Parser reports an error when the single line comment is empty #7612

- Expressions
  - Check the value of `max_allowed_packet` in the `insert` function #7528
  - Support the built-in function `json_contains` #7443
  - Support the built-in function `json_contains_path` #7596
  - Support the built-in function `encode/decode` #7622
  - Fix the issue that some time related functions are not compatible with the MySQL behaviors in some cases #7636
  - Fix the compatibility issue of parsing the time type of data in string #7654
  - Fix the issue that the time zone is not considered when computing the default value of the `DateTime` data #7655

- DML
  - Set correct `last_insert_id` in the `InsertOnDuplicateUpdate` statement #7534
  - Reduce the cases of updating the `auto_increment_id` counter #7515
  - Optimize the error message of `Duplicate Key` #7495
  - Fix the `insert...select...on duplicate key update` issue #7406
  - Support the `LOAD DATA IGNORE LINES` statement #7576

- DDL
  - Add the DDL job type and the current schema version information in the monitor #7472
  - Complete the design of the `Admin Restore Table` feature #7383
  - Fix the issue that the default value of the `Bit` type exceeds 128 #7249
  - Fix the issue that the default value of the `Bit` type cannot be `NULL` #7604
  - Reduce the interval of checking `CREATE TABLE/DATABASE` in the DDL queue #7608

  - Use the `ddl/owner/resign` HTTP interface ro release the DDL owner and start electing a new owner #7649

- TiKV Go Client
  - Support the issue that the `Seek` operation only obtains `Key` #7419

- Table Partition (Experimental)
  - Fix the issue that the `Bigint` type cannot be used as the partition key #7520
  - Support the rollback operation when an issue occurs during adding an index in the partitioned table #7437

9.3.24.2 PD

- Features
  - Support the `GetAllStores` interface #1228
  - Add the statistics of scheduling estimation in Simulator #1218

- Improvements
  - Optimize the handling process of down stores to make up replicas as soon as possible #1222
  - Optimize the start of Coordinator to reduce the unnecessary scheduling caused by restarting PD #1225
  - Optimize the memory usage to reduce the overhead caused by heartbeats #1195
  - Optimize error handling and improve the log information #1227
  - Support querying the Region information of a specific store in pd-ctl #1231
  - Support querying the topN Region information based on version comparison in pd-ctl #1233
  - Support more accurate TSO decoding in pd-ctl #1242

- Bug fix
  - Fix the issue that pd-ctl uses the `hot store` command to exit wrongly #1244

9.3.24.3 TiKV

- Performance
  - Support splitting Regions based on statistics estimation to reduce the I/O cost #3511
  - Reduce clone in the transaction scheduler #3530

- Improvements
  - Add the pushdown support for a large number of built-in functions
  - Add the `leader-transfer-max-log-lag` configuration to fix the failure issue of leader scheduling in specific scenarios #3507

  - Add the `max-open-engines` configuration to limit the number of engines opened by `tikv-importer` simultaneously #3496
  - Limit the cleanup speed of garbage data to reduce the impact on `snapshot apply` #3547
  - Broadcast the commit message for crucial Raft messages to avoid unnecessary delay #3592

- Bug fixes

  - Fix the leader election issue caused by discarding the `PreVote` message of the newly split Region #3557
  - Fix follower related statistics after merging Regions #3573
  - Fix the issue that the local reader uses obsolete Region information #3565

9.3.25 TiDB 2.1 RC1 Release Notes

On August 24, 2018, TiDB 2.1 RC1 is released! Compared with TiDB 2.1 Beta, this release has great improvement in stability, SQL optimizer, statistics information, and execution engine.

9.3.25.1 TiDB

- SQL Optimizer

  - Fix the issue that a wrong result is returned after the correlated subquery is decorrelated in some cases #6972
  - Optimize the output result of `Explain` #7011#7041
  - Optimize the choosing strategy of the outer table for `IndexJoin` #7019
  - Remove the Plan Cache of the non-`PREPARE` statement #7040
  - Fix the issue that the `INSERT` statement is not parsed and executed correctly in some cases #7068
  - Fix the issue that the `IndexJoin` result is not correct in some cases #7150
  - Fix the issue that the `NULL` value cannot be found using the unique index in some cases #7163
  - Fix the range computing issue of the prefix index in UTF-8 #7194
  - Fix the issue that result is not correct caused by eliminating the `Project` operator in some cases #7257
  - Fix the issue that `USE INDEX(PRIMARY)` cannot be used when the primary key is an integer #7316
  - Fix the issue that the index range cannot be computed using the correlated column in some cases #7357

- SQL Execution Engine

  - Fix the issue that the daylight saving time is not computed correctly in some cases #6823

  - Refactor the aggregation function framework to improve the execution efficiency of the `Stream` and `Hash` aggregation operators #6852
  - Fix the issue that the `Hash` aggregation operator cannot exit normally in some cases #6982
  - Fix the issue that `BIT_AND/BIT_OR/BIT_XOR` does not handle the non-integer data correctly #6994
  - Optimize the execution speed of the `REPLACE INTO` statement and increase the performance nearly 10 times #7027
  - Optimize the memory usage of time type data and decrease the memory usage of the time type data by fifty percent #7043
  - Fix the issue that the returned result is mixed with signed and unsigned integers in the `UNION` statement is not compatible with MySQL #7112
  - Fix the panic issue caused by the too much memory applied by `LPAD/RPAD` ↪ `/TO_BASE64/FROM_BASE64/REPEAT` #7171 #7266 #7409 #7431
  - Fix the incorrect result when `MergeJoin/IndexJoin` handles the `NULL` value #7255
  - Fix the incorrect result of `Outer Join` in some cases #7288
  - Improve the error message of `Data Truncated` to facilitate locating the wrong data and the corresponding field in the table #7401
  - Fix the incorrect result for `decimal` in some cases #7001 #7113 #7202 #7208
  - Optimize the point select performance #6937
  - Prohibit the isolation level of `Read Committed` to avoid the underlying problem #7211
  - Fix the incorrect result of `LTRIM/RTRIM/TRIM` in some cases #7291
  - Fix the issue that the `MaxOneRow` operator cannot guarantee that the returned result does not exceed one row #7375
  - Divide the Coprocessor requests with too many ranges #7454

- Statistics

  - Optimize the mechanism of statistics dynamic collection #6796
  - Fix the issue that `Auto Analyze` does not work when data is updated frequently #7022
  - Decrease the Write conflicts during the statistics dynamic update process #7124
  - Optimize the cost estimation when the statistics is incorrect #7175
  - Optimize the `AccessPath` cost estimation strategy #7233

- Server

  - Fix the bug in loading privilege information #6976
  - Fix the issue that the `Kill` command is too strict with privilege check #6954
  - Fix the issue of removing some binary numeric types #6922
  - Shorten the output log #7029
  - Handle the `mismatchClusterID` issue #7053
  - Add the `advertise-address` configuration item #7078
  - Add the `GrpcKeepAlive` option #7100
  - Add the connection or `Token` time monitor #7110

  - Optimize the data decoding performance #7149
  - Add the `PROCESSLIST` table in `INFORMMATION_SCHEMA` #7236
  - Fix the order issue when multiple rules are hit in verifying the privilege #7211
  - Change some default values of encoding related system variables to UTF-8 #7198
  - Make the slow query log show more detailed information #7302
  - Support registering tidb-server related information in PD and obtaining this information by HTTP API #7082

- Compatibility

  - Support Session variables `warning_count` and `error_count` #6945
  - Add `Scope` check when reading the system variables #6958
  - Support the `MAX_EXECUTION_TIME` syntax #7012
  - Support more statements of the `SET` syntax #7020
  - Add validity check when setting system variables #7117
  - Add the verification of the number of `PlaceHolders` in the `Prepare` statement #7162
  - Support `set character_set_results = null` #7353
  - Support the `flush status` syntax #7369
  - Fix the column size of `SET` and `ENUM` types in `information_schema` #7347
  - Support the `NATIONAL CHARACTER` syntax of statements for creating a table #7378
  - Support the `CHARACTER SET` syntax in the `LOAD DATA` statement #7391
  - Fix the column information of the `SET` and `ENUM` types #7417
  - Support the `IDENTIFIED WITH` syntax in the `CREATE USER` statement #7402
  - Fix the precision losing issue during `TIMESTAMP` computing process #7418
  - Support the validity verification of more `SYSTEM` variables #7196
  - Fix the incorrect result when the `CHAR_LENGTH` function computes the binary string #7410
  - Fix the incorrect `CONCAT` result in a statement involving `GROUP BY` #7448
  - Fix the imprecise type length issue when casting the `DECIMAL` type to the `STRING` type #7451

- DML

  - Fix the stability issue of the `Load Data` statement #6927
  - Fix the memory usage issue when performing some `Batch` operations #7086
  - Improve the performance of the `Replace Into` statement #7027
  - Fix the inconsistent precision issue when writing `CURRENT_TIMESTAMP` #7355

- DDL

  - Improve the method of DDL judging whether `Schema` is replicated to avoid misjudgement in some cases #7319
  - Fix the `SHOW CREATE TABLE` result in adding index process #6993
  - Allow the default value of `text/blob/json` to be NULL in non-restrict `sql-mode` #7230
  - Fix the `ADD INDEX` issue in some cases #7142
  - Increase the speed of adding `UNIQUE-KEY` index operation largely #7132

  - Fix the truncating issue of the prefix index in UTF-8 character set #7109
  - Add the environment variable `tidb_ddl_reorg_priority` to control the priority of the `add-index` operation #7116
  - Fix the display issue of `AUTO-INCREMENT` in `information_schema.tables` #7037
  - Support the `admin show ddl jobs <number>` command and support output specified number of DDL jobs #7028
  - Support parallel DDL job execution #6955

- Table Partition (Experimental)

  - Support top level partition
  - Support `Range Partition`

9.3.25.2 PD

- Features

  - Introduce the version control mechanism and support rolling update of the cluster with compatibility
  - Enable the `region merge` feature
  - Support the `GetPrevRegion` interface
  - Support splitting Regions in batch
  - Support storing the GC safepoint

- Improvements

  - Optimize the issue that TSO allocation is affected by the system clock going backwards
  - Optimize the performance of handling Region heartbeats
  - Optimize the Region tree performance
  - Optimize the performance of computing hotspot statistics
  - Optimize returning the error code of API interface
  - Add options of controlling scheduling strategies
  - Prohibit using special characters in `label`
  - Improve the scheduling simulator
  - Support splitting Regions using statistics in pd-ctl
  - Support formatting JSON output by calling `jq` in pd-ctl
  - Add metrics about etcd Raft state machine

- Bug fixes

  - Fix the issue that the namespace is not reloaded after switching Leader
  - Fix the issue that namespace scheduling exceeds the schedule limit
  - Fix the issue that hotspot scheduling exceeds the schedule limit
  - Fix the issue that wrong logs are output when the PD client closes
  - Fix the wrong statistics of Region heartbeat latency

9.3.25.3 TiKV

- Features
  - Support `batch split` to avoid too large Regions caused by the Write operation on hot Regions
  - Support splitting Regions based on the number of rows to improve the index scan efficiency
- Performance
  - Use `LocalReader` to separate the Read operation from the raftstore thread to lower the Read latency
  - Refactor the MVCC framework, optimize the memory usage and improve the scan Read performance
  - Support splitting Regions based on statistics estimation to reduce the I/O usage
  - Optimize the issue that the Read performance is affected by continuous Write operations on the rollback record
  - Reduce the memory usage of pushdown aggregation computing
- Improvements
  - Add the pushdown support for a large number of built-in functions and better charset support
  - Optimize the GC workflow, improve the GC speed and decrease the impact of GC on the system
  - Enable `prevote` to speed up service recovery when the network is abnormal
  - Add the related configuration items of RocksDB log files
  - Adjust the default configuration of `scheduler_latch`
  - Support setting whether to compact the data in the bottom layer of RocksDB when using tikv-ctl to compact data manually
  - Add the check for environment variables when starting TiKV
  - Support dynamically configuring the `dynamic_level_bytes` parameter based on the existing data
  - Support customizing the log format
  - Integrate tikv-fail in tikv-ctl
  - Add I/O metrics of threads
- Bug fixes
  - Fix decimal related issues
  - Fix the issue that `gRPC max_send_message_len` is set mistakenly
  - Fix the issue caused by misconfiguration of `region_size`

9.3.26 TiDB 2.1 Beta Release Notes

On June 29, 2018, TiDB 2.1 Beta is released! Compared with TiDB 2.0, this release has great improvement in stability, SQL optimizer, statistics information, and execution engine.

9.3.26.1 TiDB

- SQL Optimizer
  - Optimize the selection range of `Index Join` to improve the execution performance
  - Optimize correlated subquery, push down `Filter`, and extend the index range, to improve the efficiency of some queries by orders of magnitude
  - Support `Index Hint` and `Join Hint` in the `UPDATE` and `DELETE` statements
  - Validate Hint `TIDM_SMJ` when no available index exists
  - Support pushdown of the `ABS`, `CEIL`, `FLOOR`, `IS TRUE`, and `IS FALSE` functions
  - Handle the `IF` and `IFNULL` functions especially in the constant folding process
- SQL Execution Engine
  - Implement parallel `Hash Aggregate` operators and improve the computing performance of `Hash Aggregate` by 350% in some scenarios
  - Implement parallel `Project` operators and improve the performance by 74% in some scenarios
  - Read the data of the `Inner` table and `Outer` table of `Hash Join` concurrently to improve the execution performance
  - Fix incorrect results of `INSERT ⋯ ON DUPLICATE KEY UPDATE ⋯` in some scenarios
  - Fix incorrect results of the `CONCAT_WS`, `FLOOR`, `CEIL`, and `DIV` built-in functions
- Server
  - Add the HTTP API to scatter the distribution of table Regions in the TiKV cluster
  - Add the `auto_analyze_ratio` system variable to control the threshold value of automatic `Analyze`
  - Add the HTTP API to control whether to open the general log
  - Add the HTTP API to modify the log level online
  - Add the user information in the general log and the slow query log
  - Support the server side cursor
- Compatibility
  - Support more MySQL syntax
  - Make the `bit` aggregate function support the `ALL` parameter
  - Support the `SHOW PRIVILEGES` statement
- DML
  - Decrease the memory usage of the `INSERT INTO SELECT` statement
  - Fix the performance issue of `PlanCache`
  - Add the `tidb_retry_limit` system variable to control the automatic retry times of transactions
  - Add the `tidb_disable_txn_auto_retry` system variable to control whether the transaction tries automatically

  - Fix the accuracy issue of the written data of the `time` type
  - Support the queue of locally conflicted transactions to optimize the conflicted transaction performance
  - Fix `Affected Rows` of the `UPDATE` statement
  - Optimize the statement performance of `insert ignore on duplicate key update`

- DDL
  - Optimize the execution speed of the `CreateTable` statement
  - Optimize the execution speed of `ADD INDEX` and improve it greatly in some scenarios
  - Fix the issue that the number of added columns by `Alter table add column` exceeds the limit of the number of table columns
  - Fix the issue that DDL job retries lead to an increasing pressure on TiKV in abnormal conditions
  - Fix the issue that TiDB continuously reloads the schema information in abnormal conditions
  - Do not output the `FOREIGN KEY` related information in the result of `SHOW CREATE TABLE`
  - Support the `select tidb_is_ddl_owner()` statement to facilitate judging whether TiDB is `DDL Owner`
  - Fix the issue that the index is deleted in the `Year` type in some scenarios
  - Fix the renaming table issue in the concurrent execution scenario
  - Support the `AlterTableForce` syntax
  - Support the `AlterTableRenameIndex` syntax with `FromKey` and `ToKey`
  - Add the table name and database name in the output information of `admin show ddl jobs`

9.3.26.2 PD

- Enable Raft PreVote between PD nodes to avoid leader reelection when network recovers after network isolation
- Optimize the issue that Balance Scheduler schedules small Regions frequently
- Optimize the hotspot scheduler to improve its adaptability in traffic statistics information jitters
- Skip the Regions with a large number of rows when scheduling `region merge`
- Enable `raft learner` by default to lower the risk of unavailable data caused by machine failure during scheduling
- Remove `max-replica` from `pd-recover`
- Add `Filter` metrics
- Fix the issue that Region information is not updated after tikv-ctl unsafe recovery
- Fix the issue that TiKV disk space is used up caused by replica migration in some scenarios
- Compatibility notes

- Do not support rolling back to v2.0.x or earlier due to update of the new version storage engine
- Enable `raft learner` by default in the new version of PD. If the cluster is upgraded from 1.x to 2.1, the machine should be stopped before upgrade or a rolling update should be first applied to TiKV and then PD

9.3.26.3 TiKV

- Upgrade Rust to the `nightly-2018-06-14` version
- Enable `Raft PreVote` to avoid leader reelection generated when network recovers after network isolation
- Add a metric to display the number of files and `ingest` related information in each layer of RocksDB
- Print `key` with too many versions when GC works
- Use `static metric` to optimize multi-label metric performance (YCSB `raw get` is improved by 3%)
- Remove `box` in multiple modules and use patterns to improve the operating performance (YCSB `raw get` is improved by 3%)
- Use `asynchronous log` to improve the performance of writing logs
- Add a metric to collect the thread status
- Decease memory copy times by decreasing `box` used in the application to improve the performance

9.4 v2.0

9.4.1 TiDB 2.0.11 Release Notes

On January 03, 2019, TiDB 2.0.11 is released. The corresponding TiDB Ansible 2.0.11 is also released. Compared with TiDB 2.0.10, this release has great improvement in system compatibility and stability.

9.4.1.1 TiDB

- Fix the issue that the error is not handled properly when PD is in an abnormal condition #8764
- Fix the issue that the `Rename` operation on a table in TiDB is not compatible with that in MySQL #8809
- Fix the issue that the error message is wrongly reported when the `ADMIN CHECK TABLE` operation is performed in the process of executing the `ADD INDEX` statement #8750
- Fix the issue that the prefix index range is incorrect in some cases #8877
- Fix the panic issue of the `UPDATE` statement when columns are added in some cases #8904

9.4.1.2 TiKV

- Fix two issues about Region merge #4003, #4004

9.4.2 TiDB 2.0.10 Release Notes

On December 18, 2018, TiDB 2.0.10 is released. The corresponding TiDB Ansible 2.0.10 is also released. Compared with TiDB 2.0.9, this release has great improvement in system compatibility and stability.

9.4.2.1 TiDB

- Fix the possible issue caused by canceling a DDL job #8513
- Fix the issue that the `ORDER BY` and `UNION` clauses cannot quote the column including a table name #8514
- Fix the issue that the `UNCOMPRESS` function does not judge the incorrect input length #8607
- Fix the issue encountered by `ANSI_QUOTES SQL_MODE` when upgrading TiDB #8575
- Fix the issue that `select` returns the wrong result in some cases #8570
- Fix the possible issue that TiDB cannot exit when it receives the exit signal #8501
- Fix the issue that `IndexLookUpJoin` returns the wrong result in some cases #8508
- Avoid pushing down the filter containing `GetVar` or `SetVar` #8454
- Fix the issue that the result length of the `UNION` clauses is incorrect in some cases #8491
- Fix the issue of `PREPARE FROM @var_name` #8488
- Fix the panic issue when dumping statistics information in some cases #8464
- Fix the statistics estimation issue of point queries in some cases #8493
- Fix the panic issue when the returned default `enum` value is a string #8476
- Fix the issue that too much memory is consumed in the scenario of wide tables #8467
- Fix the issue encountered when Parser incorrectly formats the mod opcode #8431
- Fix the panic issue caused by adding foreign key constraints in some cases #8421, #8410
- Fix the issue that the `YEAR` column type incorrectly converts the zero value #8396
- Fix the panic issue occurred when the argument of the `VALUES` function is not a column #8404
- Disable Plan Cache for statements containing subqueries #8395

9.4.2.2 PD

- Fix the possible issue that RaftCluster cannot stop caused by deadlock #1370

9.4.2.3 TiKV

- Avoid transferring the leader to a newly created peer, to optimize the possible delay #3929
- Fix redundant Region heartbeats #3930

9.4.3 TiDB 2.0.9 Release Notes

On November 19, 2018, TiDB 2.0.9 is released. Compared with TiDB 2.0.8, this release has great improvement in system compatibility and stability.

9.4.3.1 TiDB

- Fix the issue caused by the empty statistics histogram #7927
- Fix the panic issue of the `UNION ALL` statement in some cases #7942
- Fix the stack overflow issue caused by wrong DDL Jobs #7959
- Add the slow log for the `Commit` operation #7983
- Fix the panic issue caused by the too large `Limit` value #8004
- Support specifying the `utf8mb4` character set in the `USING` clause #8048
- Make the `TRUNCATE` built-in function support parameters of unsigned integer type #8069
- Fix the selectivity estimation issue of the primary key for the statistics module in some cases #8150
- Add the `Session` variable to control whether `_tidb_rowid` is allowed to be written in #8126
- Fix the panic issue of `PhysicalProjection` in some cases #8154
- Fix the unstable results of the `Union` statement in some cases #8168
- Fix the issue that `NULL` is not returned by `values` in the non-`Insert` statement #8179
- Fix the issue that the statistics module cannot clear the outdated data in some cases #8184
- Make the maximum allowed running time for a transaction a configurable option 8209
- Fix the wrong comparison algorithm of `expression rewriter` in some cases #8288
- Eliminate the extra columns generated by the `UNION ORDER BY` statement #8307
- Support the `admin show next_row_id` statement #8274
- Fix the escape issue of special characters in the `Show Create Table` statement #8321
- Fix the unexpected errors in the `UNION` statement in some cases #8318
- Fix the issue that canceling a DDL job causes no rollback of a schema in some cases #8312
- Change `tidb_max_chunk_size` to a global variable #8333
- Add an upper bound to the `Scan` command of ticlient, to avoid overbound scan #8309 #8310

9.4.3.2 PD

- Fix the issue that the PD server gets stuck caused by etcd startup failure #1267
- Fix the issues related to `pd-ctl` reading the Region key #1298 #1299 #1308
- Fix the issue that the `regions/check` API returns the wrong result #1311
- Fix the issue that PD cannot restart join after a PD join failure #1279

9.4.3.3 TiKV

- Add the `end-key` limit to the `kv_scan` interface #3749
- Abandon the `max-tasks-xxx` configuration and add `max-tasks-per-worker-xxx` #3093
- Fix the `CompactFiles` issue in RocksDB #3789

9.4.4 TiDB 2.0.8 Release Notes

On October 16, 2018, TiDB 2.0.8 is released. Compared with TiDB 2.0.7, this release has great improvement in system compatibility and stability.

9.4.4.1 TiDB

- Improvement
  - Slow down the AUTO-ID increasing speed when the `Update` statement does not modify the corresponding AUTO-INCREMENT column #7846
- Bug fixes
  - Quickly create a new etcd session to recover the service when the PD leader goes down #7810
  - Fix the issue that the time zone is not considered when the default value of the `DateTime` type is calculated #7672
  - Fix the issue that `duplicate key update` inserts values incorrectly in some conditions #7685
  - Fix the issue that the predicate conditions of `UnionScan` are not pushed down #7726
  - Fix the issue that the time zone is not correctly handled when you add the `TIMESTAMP` index #7812
  - Fix the memory leak issue caused by the statistics module in some conditions #7864
  - Fix the issue that the results of `ANALYZE` cannot be obtained in some abnormal conditions #7871
  - Do not fold the function `SYSDATE`, to ensure the returned results are correct #7894
  - Fix the `substring_index` panic issue in some conditions #7896
  - Fix the issue that `OUTER JOIN` is mistakenly converted to `INNER JOIN` in some conditions #7899

9.4.4.2 TiKV

- Bug fix
  - Fix the issue that the memory consumed by Raftstore `EntryCache` keeps increasing when a node goes down #3529

9.4.5 TiDB 2.0.7 Release Notes

On September 7, 2018, TiDB 2.0.7 is released. Compared with TiDB 2.0.6, this release has great improvement in system compatibility and stability.

9.4.5.1 TiDB

- New Feature
  - Add the `PROCESSLIST` table in `information_schema` #7286
- Improvement
  - Collect more details about SQL statement execution and output the information in the `SLOW QUERY` log #7364
  - Drop the partition information in `SHOW CREATE TABLE` #7388
  - Improve the execution efficiency of the `ANALYZE` statement by setting it to the RC isolation level and low priority #7500
  - Speed up adding a unique index #7562
  - Add an option of controlling the DDL concurrency #7563
- Bug Fixes
  - Fix the issue that `USE INDEX(PRIMARY)` cannot be used in a table whose primary key is an integer #7298
  - Fix the issue that `Merge Join` and `Index Join` output incorrect results when the inner row is `NULL` #7301
  - Fix the issue that `Join` outputs an incorrect result when the chunk size is set too small #7315
  - Fix the panic issue caused by a statement of creating a table involving `range column` #7379
  - Fix the issue that `admin check table` mistakenly reports an error of a time-type column #7457
  - Fix the issue that the data with a default value `current_timestamp` cannot be queried using the `=` condition #7467
  - Fix the issue that the zero-length parameter inserted by using the `ComStmtSendLongData` command is mistakenly parsed to NULL #7508
  - Fix the issue that `auto analyze` is repeatedly executed in specific scenarios #7556
  - Fix the issue that the parser cannot parse a single line comment ended with a newline character #7635

9.4.5.2 TiKV

- Improvement
  - Open the `dynamic-level-bytes` parameter in an empty cluster by default, to reduce space amplification
- Bug Fix
  - Update `approximate size` and `approximate keys count` of a Region after Region merging

9.4.6 TiDB 2.0.6 Release Notes

On August 6, 2018, TiDB 2.0.6 is released. Compared with TiDB 2.0.5, this release has great improvement in system compatibility and stability.

9.4.6.1 TiDB

- Improvements
  - Make "set system variable" log shorter to save disk space #7031
  - Record slow operations during the execution of `ADD INDEX` in the log, to make troubleshooting easier #7083
  - Reduce transaction conflicts when updating statistics #7138
  - Improve the accuracy of row count estimation when the values pending to be estimated exceeds the statistics range #7185
  - Choose the table with a smaller estimated row count as the outer table for `Index Join` to improve its execution efficiency #7277
  - Add the recover mechanism for panics occurred during the execution of `ANALYZE TABLE`, to avoid that the tidb-server is unavailable caused by abnormal behavior in the process of collecting statistics #7228
  - Return `NULL` and the corresponding warning when the results of `RPAD`/`LPAD` exceed the value of the `max_allowed_packet` system variable, compatible with MySQL #7244
  - Set the upper limit of placeholders count in the `PREPARE` statement to 65535, compatible with MySQL #7250
- Bug Fixes
  - Fix the issue that the `DROP USER` statement is incompatible with MySQL behavior in some cases #7014
  - Fix the issue that statements like `INSERT`/`LOAD DATA` meet OOM after opening `tidb_batch_insert` #7092
  - Fix the issue that the statistics fail to automatically update when the data of a table keeps updating #7093

- Fix the issue that the firewall breaks inactive gPRC connections #7099
- Fix the issue that prefix index returns a wrong result in some scenarios #7126
- Fix the panic issue caused by outdated statistics in some scenarios #7155
- Fix the issue that one piece of index data is missed after the `ADD INDEX` operation in some scenarios #7156
- Fix the wrong result issue when querying `NULL` values using the unique index in some scenarios #7172
- Fix the messy code issue of the `DECIMAL` multiplication result in some scenarios #7212
- Fix the wrong result issue of `DECIMAL` modulo operation in some scenarios #7245
- Fix the issue that the `UPDATE/DELETE` statement in a transaction returns a wrong result under some special sequence of statements #7219
- Fix the panic issue of the `UNION ALL/UPDATE` statement during the process of building the execution plan in some scenarios #7225
- Fix the issue that the range of prefix index is calculated incorrectly in some scenarios #7231
- Fix the issue that the `LOAD DATA` statement fails to write the binlog in some scenarios #7242
- Fix the wrong result issue of `SHOW CREATE TABLE` during the execution process of `ADD INDEX` in some scenarios #7243
- Fix the issue that panic occurs when `Index Join` does not initialize timestamps in some scenarios #7246
- Fix the false alarm issue when `ADMIN CHECK TABLE` mistakenly uses the timezone in the session #7258
- Fix the issue that `ADMIN CLEANUP INDEX` does not clean up the index in some scenarios #7265
- Disable the Read Committed isolation level #7282

9.4.6.2 TiKV

- Improvements
  - Enlarge scheduler's default slots to reduce false conflicts
  - Reduce continuous records of rollback transactions, to improve the Read performance when conflicts are extremely severe
  - Limit the size and number of RocksDB log files, to reduce unnecessary disk usage in long-running condition
- Bug Fixes
  - Fix the crash issue when converting the data type from string to decimal

9.4.7 TiDB 2.0.5 Release Notes

On July 6, 2018, TiDB 2.0.5 is released. Compared with TiDB 2.0.4, this release has great improvement in system compatibility and stability.

9.4.7.1 TiDB

- New Features
  - Add the `tidb_disable_txn_auto_retry` system variable which is used to disable the automatic retry of transactions #6877
- Improvements
  - Optimize the cost calculation of `Selection` to make the result more accurate #6989
  - Select the query condition that completely matches the unique index or the primary key as the query path directly #6966
  - Execute necessary cleanup when failing to start the service #6964
  - Handle `\N` as NULL in the `Load Data` statement #6962
  - Optimize the code structure of CBO #6953
  - Report the monitoring metrics earlier when starting the service #6931
  - Optimize the format of slow queries by removing the line breaks in SQL statements and adding user information #6920
  - Support multiple asterisks in comments #6858
- Bug Fixes
  - Fix the issue that `KILL QUERY` always requires SUPER privilege #7003
  - Fix the issue that users might fail to login when the number of users exceeds 1024 #6986
  - Fix an issue about inserting unsigned `float`/`double` data #6940
  - Fix the compatibility of the `COM_FIELD_LIST` command to resolve the panic issue in some MariaDB clients #6929
  - Fix the `CREATE TABLE IF NOT EXISTS LIKE` behavior #6928
  - Fix an issue in the process of TopN pushdown #6923
  - Fix the ID record issue of the currently processing row when an error occurs in executing `Add Index` #6903

9.4.7.2 PD

- Fix the issue that replicas migration uses up TiKV disks space in some scenarios
- Fix the crash issue caused by `AdjacentRegionScheduler`

9.4.7.3 TiKV

- Fix the potential overflow issue in decimal operations
- Fix the dirty read issue that might occur in the process of merging

9.4.8 TiDB 2.0.4 Release Notes

On June 15, 2018, TiDB 2.0.4 is released. Compared with TiDB 2.0.3, this release has great improvement in system compatibility and stability.

9.4.8.1 TiDB

- Support the `ALTER TABLE t DROP COLUMN a CASCADE` syntax
- Support configuring the value of `tidb_snapshot` to TSO
- Refine the display of statement types in monitoring items
- Optimize the accuracy of query cost estimation
- Configure the `backoff max delay` parameter of gRPC
- Support configuring the memory threshold of a single statement in the configuration file
- Refactor the error of Optimizer
- Fix the side effects of the `Cast Decimal` data
- Fix the wrong result issue of the `Merge Join` operator in specific scenarios
- Fix the issue of converting the Null object to String
- Fix the issue of casting the JSON type of data to the JSON type
- Fix the issue that the result order is not consistent with MySQL in the condition of `Union` + `OrderBy`
- Fix the compliance rules issue when the `Union` statement checks the `Limit/OrderBy` clause
- Fix the compatibility issue of the `Union All` result
- Fix a bug in predicate pushdown
- Fix the compatibility issue of the `Union` statement with the `For Update` clause
- Fix the issue that the `concat_ws` function mistakenly truncates the result

9.4.8.2 PD

- Improve the behavior of the unset scheduling argument `max-pending-peer-count` by changing it to no limit for the maximum number of `PendingPeer`s

9.4.8.3 TiKV

- Add the RocksDB `PerfContext` interface for debugging
- Remove the `import-mode` parameter
- Add the `region-properties` command for `tikv-ctl`
- Fix the issue that `reverse-seek` is slow when many RocksDB tombstones exist
- Fix the crash issue caused by `do_sub`
- Make GC record the log when GC encounters many versions of data

9.4.9 TiDB 2.0.3 Release Notes

On June 1, 2018, TiDB 2.0.3 is released. Compared with TiDB 2.0.2, this release has great improvement in system compatibility and stability.

9.4.9.1 TiDB

- Support modifying the log level online
- Support the `COM_CHANGE_USER` command
- Support using the `TIME` type parameters under the binary protocol
- Optimize the cost estimation of query conditions with the `BETWEEN` expression
- Do not display the `FOREIGN KEY` information in the result of `SHOW CREATE TABLE`
- Optimize the cost estimation for queries with the `LIMIT` clause
- Fix the issue about the `YEAR` type as the unique index
- Fix the issue about `ON DUPLICATE KEY UPDATE` in conditions without the unique index
- Fix the compatibility issue of the `CEIL` function
- Fix the accuracy issue of the `DIV` calculation in the `DECIMAL` type
- Fix the false alarm of `ADMIN CHECK TABLE`
- Fix the panic issue of `MAX/MIN` under specific expression parameters
- Fix the issue that the result of `JOIN` is null in special conditions
- Fix the `IN` expression issue when building and querying Range
- Fix a Range calculation issue when using `Prepare` to query and `Plan Cache` is enabled
- Fix the issue that the Schema information is frequently loaded in abnormal conditions

9.4.9.2 PD

- Fix the panic issue when collecting hot-cache metrics in specific conditions
- Fix the issue about scheduling of the obsolete Regions

9.4.9.3 TiKV

- Fix the bug that the learner flag mistakenly reports to PD
- Report an error instead of getting a result if `divisor/dividend` is 0 in `do_div_mod`

9.4.10 TiDB 2.0.2 Release Notes

On May 21, 2018, TiDB 2.0.2 is released. Compared with TiDB 2.0.1, this release has great improvement in system stability.

9.4.10.1 TiDB

- Fix the issue of pushing down the Decimal division expression
- Support using the `USE INDEX` syntax in the `Delete` statement
- Forbid using the `shard_row_id_bits` feature in columns with `Auto-Increment`
- Add the timeout mechanism for writing Binlog

9.4.10.2 PD

- Make the balance leader scheduler filter the disconnected nodes
- Modify the timeout of the transfer leader operator to 10s
- Fix the issue that the label scheduler does not schedule when the cluster Regions are in an unhealthy state
- Fix the improper scheduling issue of `evict leader scheduler`

9.4.10.3 TiKV

- Fix the issue that the Raft log is not printed
- Support configuring more gRPC related parameters
- Support configuring the timeout range of leader election
- Fix the issue that the obsolete learner is not deleted
- Fix the issue that the snapshot intermediate file is mistakenly deleted

9.4.11 TiDB 2.0.1 Release Notes

On May 16, 2018, TiDB 2.0.1 is released. Compared with TiDB 2.0.0 (GA), this release has great improvement in MySQL compatibility and system stability.

9.4.11.1 TiDB

- Update the progress of `Add Index` to the DDL job information in real time
- Add the `tidb_auto_analyze_ratio` session variable to control the threshold value of automatic statistics update
- Fix an issue that not all residual states are cleaned up when the transaction commit fails
- Fix a bug about adding indexes in some conditions
- Fix the correctness related issue when DDL modifies surface operations in some concurrent scenarios
- Fix a bug that the result of `LIMIT` is incorrect in some conditions
- Fix a capitalization issue of the `ADMIN CHECK INDEX` statement to make its index name case insensitive
- Fix a compatibility issue of the `UNION` statement
- Fix a compatibility issue when inserting data of `TIME` type
- Fix a goroutine leak issue caused by `copIteratorTaskSender` in some conditions
- Add an option for TiDB to control the behaviour of Binlog failure
- Refactor the `Coprocessor` slow log to distinguish between the scenario of tasks with long processing time and long waiting time
- Log nothing when meeting MySQL protocol handshake error, to avoid too many logs caused by the load balancer Keep Alive mechanism
- Refine the “Out of range value for column” error message

- Fix a bug when there is a subquery in an `Update` statement
- Change the behaviour of handling `SIGTERM`, and do not wait for all queries to terminate anymore

9.4.11.2 PD

- Add the `Scatter Range` scheduler to balance Regions with the specified key range
- Optimize the scheduling of Merge Region to prevent the newly split Region from being merged
- Add Learner related metrics
- Fix the issue that the scheduler is mistakenly deleted after restart
- Fix the error that occurs when parsing the configuration file
- Fix the issue that the etcd leader and the PD leader are not replicated
- Fix the issue that Learner still appears after it is closed
- Fix the issue that Regions fail to load because the packet size is too large

9.4.11.3 TiKV

- Fix the issue that `SELECT FOR UPDATE` prevents others from reading
- Optimize the slow query log
- Reduce the number of `thread_yield` calls
- Fix the bug that raftstore is accidentally blocked when generating the snapshot
- Fix the issue that Learner cannot be successfully elected in special conditions
- Fix the issue that split might cause dirty read in extreme conditions
- Correct the default value of the read thread pool configuration
- Speed up Delete Range

9.4.12 TiDB 2.0 Release Notes

On April 27, 2018, TiDB 2.0 GA is released! Compared with TiDB 1.0, this release has great improvement in MySQL compatibility, SQL optimizer, executor, and stability.

9.4.12.1 TiDB

- SQL Optimizer
  - Use more compact data structure to reduce the memory usage of statistics information
  - Speed up loading statistics information when starting a tidb-server process
  - Support updating statistics information dynamically [experimental]
  - Optimize the cost model to provide more accurate query cost evaluation
  - Use `Count-Min Sketch` to estimate the cost of point queries more accurately
  - Support analyzing more complex conditions to make full use of indexes

  - Support manually specifying the `Join` order using the `STRAIGHT_JOIN` syntax
  - Use the Stream Aggregation operator when the `GROUP BY` clause is empty to improve the performance
  - Support using indexes for the `MAX/MIN` function
  - Optimize the processing algorithms for correlated subqueries to support decorrelating more types of correlated subqueries and transform them to `Left Outer Join`
  - Extend `IndexLookupJoin` to be used in matching the index prefix

- SQL Execution Engine
  - Refactor all operators using the Chunk architecture, improve the execution performance of analytical queries, and reduce memory usage. There is a significant improvement in the TPC-H benchmark result.
  - Support the Streaming Aggregation operators pushdown
  - Optimize the `Insert Into Ignore` statement to improve the performance by over 10 times
  - Optimize the `Insert On Duplicate Key Update` statement to improve the performance by over 10 times
  - Optimize `Load Data` to improve the performance by over 10 times
  - Push down more data types and functions to TiKV
  - Support computing the memory usage of physical operators, and specifying the processing behavior in the configuration file and system variables when the memory usage exceeds the threshold
  - Support limiting the memory usage by a single SQL statement to reduce the risk of OOM
  - Support using implicit RowID in CRUD operations
  - Improve the performance of point queries

- Server
  - Support the Proxy Protocol
  - Add more monitoring metrics and refine the log
  - Support validating the configuration files
  - Support obtaining the information of TiDB parameters through HTTP API
  - Resolve Lock in the Batch mode to speed up garbage collection
  - Support multi-threaded garbage collection
  - Support TLS

- Compatibility
  - Support more MySQL syntaxes
  - Support modifying the `lower_case_table_names` system variable in the configuration file to support the OGG data replication tool
  - Improve compatibility with the Navicat management tool
  - Support displaying the table creating time in `Information_Schema`
  - Fix the issue that the return types of some functions/expressions differ from MySQL

  - Improve compatibility with JDBC
  - Support more SQL Modes

- DDL

  - Optimize the `Add Index` operation to greatly improve the execution speed in some scenarios
  - Attach a lower priority to the `Add Index` operation to reduce the impact on online business
  - Output more detailed status information of the DDL jobs in `Admin Show DDL Jobs`
  - Support querying the original statements of currently running DDL jobs using `Admin Show DDL Job Queries JobID`
  - Support recovering the index data using `Admin Recover Index` for disaster recovery
  - Support modifying Table Options using the `Alter` statement

9.4.12.2 PD

- Support `Region Merge`, to merge empty Regions after deleting data [experimental]
- Support `Raft Learner` [experimental]
- Optimize the scheduler

  - Make the scheduler to adapt to different Region sizes
  - Improve the priority and speed of restoring data during TiKV outage
  - Speed up data transferring when removing a TiKV node
  - Optimize the scheduling policies to prevent the disks from becoming full when the space of TiKV nodes is insufficient
  - Improve the scheduling efficiency of the balance-leader scheduler
  - Reduce the scheduling overhead of the balance-region scheduler
  - Optimize the execution efficiency of the the hot-region scheduler

- Operations interface and configuration

  - Support TLS
  - Support prioritizing the PD leaders
  - Support configuring the scheduling policies based on labels
  - Support configuring stores with a specific label not to schedule the Raft leader
  - Support splitting Region manually to handle the hotspot in a single Region
  - Support scattering a specified Region to manually adjust Region distribution in some cases
  - Add check rules for configuration parameters and improve validity check of the configuration items

- Debugging interface

  - Add the `Drop Region` debugging interface

  - Add the interfaces to enumerate the health status of each PD

- Statistics
  - Add statistics about abnormal Regions
  - Add statistics about Region isolation level
  - Add scheduling related metrics

- Performance
  - Keep the PD leader and the etcd leader together in the same node to improve write performance
  - Optimize the performance of Region heartbeat

9.4.12.3 TiKV

- Features
  - Protect critical configuration from incorrect modification
  - Support `Region Merge` [experimental]
  - Add the `Raw DeleteRange` API
  - Add the `GetMetric` API
  - Add `Raw Batch Put`, `Raw Batch Get`, `Raw Batch Delete` and `Raw Batch Scan`
  - Add Column Family options for the RawKV API and support executing operation on a specific Column Family
  - Support Streaming and Streaming Aggregation in Coprocessor
  - Support configuring the request timeout of Coprocessor
  - Carry timestamps with Region heartbeats
  - Support modifying some RocksDB parameters online, such as `block-cache-size`
  - Support configuring the behavior of Coprocessor when it encounters some warnings or errors
  - Support starting in the importing data mode to reduce write amplification during the data importing process
  - Support manually splitting Region in halves
  - Improve the data recovery tool `tikv-ctl`
  - Return more statistics in Coprocessor to guide the behavior of TiDB
  - Support the `ImportSST` API to import SST files [experimental]
  - Add the TiKV Importer binary to integrate with TiDB Lightning to import data quickly [experimental]

- Performance
  - Optimize read performance using `ReadPool` and increase the `raw_get/get/batch_get` by 30%
  - Improve metrics performance
  - Inform PD immediately once the Raft snapshot process is completed to speed up balancing

  - Solve performance jitter caused by RocksDB flushing
  - Optimize the space reclaiming mechanism after deleting data
  - Speed up garbage cleaning while starting the server
  - Reduce the I/O overhead during replica migration using `DeleteFilesInRanges`

- Stability

  - Fix the issue that gRPC call does not get returned when the PD leader switches
  - Fix the issue that it is slow to offline nodes caused by snapshots
  - Limit the temporary space usage consumed by migrating replicas
  - Report the Regions that cannot elect a leader for a long time
  - Update the Region size information in time according to compaction events
  - Limit the size of scan lock to avoid request timeout
  - Limit the memory usage when receiving snapshots to avoid OOM
  - Increase the speed of CI test
  - Fix the OOM issue caused by too many snapshots
  - Configure `keepalive` of gRPC
  - Fix the OOM issue caused by an increase of the Region number

9.4.12.4 TiSpark

TiSpark uses a separate version number. The current TiSpark version is 1.0 GA. The components of TiSpark 1.0 provide distributed computing of TiDB data using Apache Spark.

- Provide a gRPC communication framework to read data from TiKV
- Provide encoding and decoding of TiKV component data and communication protocol
- Provide calculation pushdown, which includes:

  - Aggregate pushdown
  - Predicate pushdown
  - TopN pushdown
  - Limit pushdown

- Provide index related support

  - Transform predicate into Region key range or secondary index
  - Optimize `Index Only` queries - Adaptively downgrade index scan to table scan per Region

- Provide cost-based optimization

  - Support statistics
  - Select index
  - Estimate broadcast table cost

- Provide support for multiple Spark interfaces

  - Support Spark Shell
  - Support ThriftServer/JDBC

- Support Spark-SQL interaction
- Support PySpark Shell
- Support SparkR

9.4.13 TiDB 2.0 RC5 Release Notes

On April 17, 2018, TiDB 2.0 RC5 is released. This release has great improvement in MySQL compatibility, SQL optimization and stability.

9.4.13.1 TiDB

- Fix the issue about applying the `Top-N` pushdown rule
- Fix the estimation of the number of rows for the columns that contain NULL values
- Fix the zero value of the Binary type
- Fix the `BatchGet` issue within a transaction
- Clean up the written data while rolling back the `Add Index` operation, to reduce consumed space
- Optimize the `insert on duplicate key update` statement to improve the performance by 10 times
- Fix the issue about the type of the results returned by the `UNIX_TIMESTAMP` function
- Fix the issue that the NULL value is inserted while adding NOT NULL columns
- Support showing memory usage of the executing statements in the `Show Process List` statement
- Fix the issue that `Alter Table Modify Column` reports an error in extreme conditions
- Support setting the table comment using the `Alter` statement

9.4.13.2 PD

- Add support for Raft Learner
- Optimize the Balance Region Scheduler to reduce scheduling overhead
- Adjust the default value of `schedule-limit` configuration
- Fix the issue of allocating ID frequently
- Fix the compatibility issue when adding a new scheduler

9.4.13.3 TiKV

- Support the Region specified by `compact` in `tikv-ctl`
- Support Batch Put, Batch Get, Batch Delete and Batch Scan in the RawKVClient
- Fix the OOM issue caused by too many snapshots
- Return more detailed error information in Coprocessor
- Support dynamically modifying the `block-cache-size` in TiKV through `tikv-ctl`
- Further improve `importer`
- Simplify the `ImportSST::Upload` interface

- Configure the `keepalive` property of gRPC
- Split `tikv-importer` from TiKV as an independent binary
- Provide statistics about the number of rows scanned by each `scan range` in Coprocessor
- Fix the compilation issue on the macOS system
- Fix the issue of misusing a RocksDB metric
- Support the `overflow as warning` option in Coprocessor

9.4.14 TiDB 2.0 RC4 Release Notes

On March 30, 2018, TiDB 2.0 RC4 is released. This release has great improvement in MySQL compatibility, SQL optimization and stability.

9.4.14.1 TiDB

- Support `SHOW GRANTS FOR CURRENT_USER();`
- Fix the issue that the `Expression` in `UnionScan` is not cloned
- Support the `SET TRANSACTION` syntax
- Fix the potential goroutine leak issue in `copIterator`
- Fix the issue that `admin check table` misjudges the unique index including null
- Support displaying floating point numbers using scientific notation
- Fix the type inference issue during binary literal computing
- Fix the issue in parsing the `CREATE VIEW` statement
- Fix the panic issue when one statement contains both `ORDER BY` and `LIMIT 0`
- Improve the execution performance of `DecodeBytes`
- Optimize `LIMIT 0` to `TableDual`, to avoid building useless execution plans

9.4.14.2 PD

- Support splitting Region manually to handle the hot spot in a single Region
- Fix the issue that the label property is not displayed when `pdctl` runs `config show all`
- Optimize metrics and code structure

9.4.14.3 TiKV

- Limit the memory usage during receiving snapshots, to avoid OOM in extreme conditions
- Support configuring the behavior of Coprocessor when it encounters warnings
- Support importing the data pattern in TiKV
- Support splitting Region in the middle
- Increase the speed of CI test
- Use `crossbeam channel`
- Fix the issue that too many logs are output caused by leader missing when TiKV is isolated

9.4.15 TiDB 2.0 RC3 Release Notes

On March 23, 2018, TiDB 2.0 RC3 is released. This release has great improvement in MySQL compatibility, SQL optimization and stability.

9.4.15.1 TiDB

- Fix the wrong result issue of `MAX/MIN` in some scenarios
- Fix the issue that the result of `Sort Merge Join` does not show in order of Join Key in some scenarios
- Fix the error of comparison between `uint` and `int` in boundary conditions
- Optimize checks on length and precision of the floating point type, to improve compatibility with MySQL
- Improve the parsing error log of time type and add more error information
- Improve memory control and add statistics about `IndexLookupExecutor` memory
- Optimize the execution speed of `ADD INDEX` to greatly increase the speed in some scenarios
- Use the Stream Aggregation operator when the `GROUP BY` substatement is empty, to increase the speed
- Support closing the `Join Reorder` optimization in the optimizer using `STRAIGHT_JOIN`
- Output more detailed status information of DDL jobs in `ADMIN SHOW DDL JOBS`
- Support querying the original statements of currently running DDL jobs using `ADMIN SHOW DDL JOB QUERIES`
- Support recovering the index data using `ADMIN RECOVER INDEX` for disaster recovery
- Attach a lower priority to the `ADD INDEX` operation to reduce the impact on online business
- Support aggregation functions with JSON type parameters, such as `SUM/AVG`
- Support modifying the `lower_case_table_names` system variable in the configuration file, to support the OGG data replication tool
- Improve compatibility with the Navicat management tool
- Support using implicit RowID in CRUD operations

9.4.15.2 PD

- Support Region Merge, to merge empty Regions or small Regions after deleting data
- Ignore the nodes that have a lot of pending peers during adding replicas, to improve the speed of restoring replicas or making nodes offline
- Fix the frequent scheduling issue caused by a large number of empty Regions
- Optimize the scheduling speed of leader balance in scenarios of unbalanced resources within different labels
- Add more statistics about abnormal Regions

9.4.15.3 TiKV

- Support Region Merge
- Inform PD immediately once the Raft snapshot process is completed, to speed up balancing
- Add the Raw DeleteRange API
- Add the GetMetric API
- Reduce the I/O fluctuation caused by RocksDB sync files
- Optimize the space reclaiming mechanism after deleting data
- Improve the data recovery tool `tikv-ctl`
- Fix the issue that it is slow to make nodes down caused by snapshot
- Support streaming in Coprocessor
- Support Readpool and increase the `raw_get/get/batch_get` by 30%
- Support configuring the request timeout of Coprocessor
- Support streaming aggregation in Coprocessor
- Carry time information in Region heartbeats
- Limit the space usage of snapshot files to avoid consuming too much disk space
- Record and report the Regions that cannot elect a leader for a long time
- Speed up garbage cleaning when starting the server
- Update the size information about the corresponding Region according to compaction events
- Limit the size of `scan lock` to avoid request timeout
- Use `DeleteRange` to speed up Region deletion
- Support modifying RocksDB parameters online

9.4.16 TiDB 2.0 RC1 Release Notes

On March 9, 2018, TiDB 2.0 RC1 is released. This release has great improvement in MySQL compatibility, SQL optimization and stability.

9.4.16.1 TiDB

- Support limiting the memory usage by a single SQL statement, to reduce the risk of OOM
- Support pushing the Stream Aggregate operator down to TiKV
- Support validating the configuration file
- Support obtaining the information of TiDB configuration through HTTP API
- Compatible with more MySQL syntax in Parser
- Improve the compatibility with Navicat
- Improve the optimizer and extract common expressions with multiple OR conditions, to choose better query plan
- Improve the optimizer and convert subqueries to Join operators in more scenarios, to choose better query plan
- Resolve Lock in the Batch mode to increase the garbage collection speed

- Fix the length of Boolean field to improve compatibility
- Optimize the Add Index operation and give lower priority to all write and read operations, to reduce the impact on online business

9.4.16.2 PD

- Optimize the logic of code used to check the Region status to improve performance
- Optimize the output of log information in abnormal conditions to facilitate debugging
- Fix the monitor statistics that the disk space of TiKV nodes is not enough
- Fix the wrong reporting issue of the health interface when TLS is enabled
- Fix the issue that concurrent addition of replicas might exceed the threshold value of configuration, to improve stability

9.4.16.3 TiKV

- Fix the issue that gRPC call is not cancelled when PD leaders switch
- Protect important configuration which cannot be changed after initial configuration
- Add gRPC APIs used to obtain metrics
- Check whether SSD is used when you start the cluster
- Optimize the read performance using ReadPool, and improve the performance by 30% in the `raw get` test
- Improve metrics and optimize the usage of metrics

9.4.17 TiDB 1.1 Beta Release Notes

On February 24, 2018, TiDB 1.1 Beta is released. This release has great improvement in MySQL compatibility, SQL optimization, stability, and performance.

9.4.17.1 TiDB

- Add more monitoring metrics and refine the log
- Compatible with more MySQL syntax
- Support displaying the table creating time in `information_schema`
- Optimize queries containing the `MaxOneRow` operator
- Configure the size of intermediate result sets generated by Join, to further reduce the memory used by Join
- Add the `tidb_config` session variable to output the current TiDB configuration
- Fix the panic issue in the `Union` and `Index Join` operators
- Fix the wrong result issue of the `Sort Merge Join` operator in some scenarios
- Fix the issue that the `Show Index` statement shows indexes that are in the process of adding
- Fix the failure of the `Drop Stats` statement

- Optimize the query performance of the SQL engine to improve the test result of the Sysbench Select/OLTP by 10%
- Improve the computing speed of subqueries in the optimizer using the new execution engine; compared with TiDB 1.0, TiDB 1.1 Beta has great improvement in tests like TPC-H and TPC-DS

9.4.17.2 PD

- Add the Drop Region debug interface
- Support setting priority of the PD leader
- Support configuring stores with a specific label not to schedule Raft leaders
- Add the interfaces to enumerate the health status of each PD
- Add more metrics
- Keep the PD leader and the etcd leader together as much as possible in the same node
- Improve the priority and speed of restoring data when TiKV goes down
- Enhance the validity check of the `data-dir` configuration item
- Optimize the performance of Region heartbeat
- Fix the issue that hot spot scheduling violates label constraint
- Fix other stability issues

9.4.17.3 TiKV

- Traverse locks using offset + limit to avoid potential GC problems
- Support resolving locks in batches to improve GC speed
- Support GC concurrency to improve GC speed
- Update the Region size using the RocksDB compaction listener for more accurate PD scheduling
- Delete the outdated data in batches using `DeleteFilesInRanges`, to make TiKV start faster
- Configure the Raft snapshot max size to avoid the retained files taking up too much space
- Support more recovery operations in `tikv-ctl`
- Optimize the ordered flow aggregation operation
- Improve metrics and fix bugs

9.4.18 TiDB 1.1 Alpha Release Notes

On January 19, 2018, TiDB 1.1 Alpha is released. This release has great improvement in MySQL compatibility, SQL optimization, stability, and performance.

9.4.18.1 TiDB

- SQL parser

  - Support more syntax

- SQL query optimizer
  - Use more compact structure to reduce statistics info memory usage
  - Speed up loading statistics info when starting tidb-server
  - Provide more accurate query cost evaluation
  - Use `Count-Min Sketch` to estimate the cost of queries using unique index more accurately
  - Support more complex conditions to make full use of index
- SQL executor
  - Refactor all executor operators using Chunk architecture, improve the execution performance of analytical statements and reduce memory usage
  - Optimize performance of the `INSERT IGNORE` statement
  - Push down more types and functions to TiKV
  - Support more `SQL_MODE`
  - Optimize the `Load Data` performance to increase the speed by 10 times
  - Optimize the `Use Database` performance
  - Support statistics on the memory usage of physical operators
- Server
  - Support the PROXY protocol

9.4.18.2 PD

- Add more APIs
- Support TLS
- Add more cases for scheduling Simulator
- Schedule to adapt to different Region sizes
- Fix some bugs about scheduling

9.4.18.3 TiKV

- Support Raft learner
- Optimize Raft Snapshot and reduce the I/O overhead
- Support TLS
- Optimize the RocksDB configuration to improve performance
- Optimize `count (*)` and query performance of unique index in Coprocessor
- Add more failpoints and stability test cases
- Solve the reconnection issue between PD and TiKV
- Enhance the features of the data recovery tool `tikv-ctl`
- Support splitting according to table in Region
- Support the `Delete Range` feature
- Support setting the I/O limit caused by snapshot
- Improve the flow control mechanism

9.5 v1.0

9.5.1 TiDB 1.0.8 Release Notes

On February 11, 2018, TiDB 1.0.8 is released with the following updates:

9.5.1.1 TiDB

- Fix issues in the `Outer Join` result in some scenarios
- Optimize the performance of the `InsertIntoIgnore` statement
- Fix the issue in the `ShardRowID` option
- Add limitation (Configurable, the default value is 5000) to the DML statements number within a transaction
- Fix an issue in the Table/Column aliases returned by the `Prepare` statement
- Fix an issue in updating statistics delta
- Fix a panic error in the `Drop Column` statement
- Fix an DML issue when running the `Add Column After` statement
- Improve the stability of the GC process by ignoring the regions with GC errors
- Run GC concurrently to accelerate the GC process
- Provide syntax support for the `CREATE INDEX` statement

9.5.1.2 PD

- Reduce the lock overheat of the region heartbeats
- Fix the issue that a hot region scheduler selects the wrong Leader

9.5.1.3 TiKV

- Use `DeleteFilesInRanges` to clear stale data and improve the TiKV starting speed
- Using `Decimal` in Coprocessor sum
- Sync the metadata of the received Snapshot compulsorily to ensure its safety

To upgrade from 1.0.7 to 1.0.8, follow the rolling upgrade order of PD -> TiKV -> TiDB.

9.5.2 TiDB 1.0.7 Release Notes

On January 22, 2018, TiDB 1.0.7 is released with the following updates:

9.5.2.1 TiDB

- Optimize the `FIELD_LIST` command
- Fix data race of the information schema
- Avoid adding read-only statements to history
- Add the `session` variable to control the log query
- Fix the resource leak issue in statistics
- Fix the goroutine leak issue
- Add schema info API for the http status server
- Fix an issue about `IndexJoin`
- Update the behavior when `RunWorker` is false in DDL
- Improve the stability of test results in statistics
- Support `PACK_KEYS` syntax for the `CREATE TABLE` statement
- Add `row_id` column for the null pushdown schema to optimize performance

9.5.2.2 PD

- Fix possible scheduling loss issue in abnormal conditions
- Fix the compatibility issue with proto3
- Add the log

9.5.2.3 TiKV

- Support `Table Scan`
- Support the remote mode in tikv-ctl
- Fix the format compatibility issue of tikv-ctl proto
- Fix the loss of scheduling command from PD
- Add timeout in Push metric

To upgrade from 1.0.6 to 1.0.7, follow the rolling upgrade order of PD -> TiKV -> TiDB.

9.5.3 TiDB 1.0.6 Release Notes

On January 08, 2018, TiDB 1.0.6 is released with the following updates:

9.5.3.1 TiDB

- Support the `Alter Table Auto_Increment` syntax
- Fix the bug in Cost Based computation and the `Null Json` issue in statistics
- Support the extension syntax to shard the implicit row ID to avoid write hot spot for a single table
- Fix a potential DDL issue
- Consider the timezone setting in the `curtime`, `sysdate` and `curdate` functions
- Support the `SEPARATOR` syntax in the `GROUP_CONCAT` function
- Fix the wrong return type issue of the `GROUP_CONCAT` function.

9.5.3.2 PD

- Fix store selection problem of hot-region scheduler

9.5.3.3 TiKV

None.

To upgrade from 1.0.5 to 1.0.6, follow the rolling upgrade order of PD -> TiKV -> TiDB.

9.5.4 TiDB 1.0.5 Release Notes

On December 26, 2017, TiDB 1.0.5 is released with the following updates:

9.5.4.1 TiDB

- Add the max value for the current Auto_Increment ID in the `Show Create Table` statement.
- Fix a potential goroutine leak.
- Support outputting slow queries into a separate file.
- Load the `TimeZone` variable from TiKV when creating a new session.
- Support the schema state check so that the `Show Create Table`and `Analyze` statements process the public table/index only.
- The `set transaction read only` should affect the `tx_read_only` variable.
- Clean up incremental statistic data when rolling back.
- Fix the issue of missing index length in the `Show Create Table` statement.

9.5.4.2 PD

- Fix the issue that the leaders stop balancing under some circumstances.
  - 869
  - 874
- Fix potential panic during bootstrapping.

9.5.4.3 TiKV

- Fix the issue that it is slow to get the CPU ID using the `get_cpuid` function.
- Support the `dynamic-level-bytes` parameter to improve the space collection situation.

To upgrade from 1.0.4 to 1.0.5, follow the rolling upgrade order of PD -> TiKV -> TiDB.

9.5.5 TiDB 1.0.4 Release Notes

On December 11, 2017, TiDB 1.0.4 is released with the following updates:

9.5.5.1 TiDB

- Speed up the loading of the statistics when starting the `tidb-server`
- Improve the performance of the `show variables` statement
- Fix a potential issue when using the `Add Index` statement to handle the combined indexes
- Fix a potential issue when using the `Rename Table` statement to move a table to another database
- Accelerate the effectiveness for the `Alter/Drop User` statement

9.5.5.2 TiKV

- Fix a possible performance issue when a snapshot is applied
- Fix the performance issue for reverse scan after removing a lot of data
- Fix the wrong encoded result for the Decimal type under special circumstances

To upgrade from 1.0.3 to 1.0.4, follow the rolling upgrade order of PD -> TiKV -> TiDB.

9.5.6 TiDB 1.0.3 Release Notes

On November 28, 2017, TiDB 1.0.3 is released with the following updates:

9.5.6.1 TiDB

- Optimize the performance in transaction conflicts scenario
- Add the `TokenLimit` option in the config file
- Output the default database in slow query logs
- Remove the DDL statement from query duration metrics
- Optimize the query cost estimation
- Fix the index prefix issue when creating tables
- Support pushing down the expressions for the Float type to TiKV
- Fix the issue that it is slow to add index for tables with discrete integer primary index
- Reduce the unnecessary statistics updates
- Fix a potential issue during the transaction retry

9.5.6.2 PD

- Support adding more types of schedulers using API

9.5.6.3 TiKV

- Fix the deadlock issue with the PD client
- Fix the issue that the wrong leader value is prompted for `NotLeader`
- Fix the issue that the chunk size is too large in the coprocessor

To upgrade from 1.0.2 to 1.0.3, follow the rolling upgrade order of PD -> TiKV -> TiDB.

9.5.7 TiDB 1.0.2 Release Notes

On November 13, 2017, TiDB 1.0.2 is released with the following updates:

9.5.7.1 TiDB

- Optimize the cost estimation of index point query
- Support the `Alter Table Add Column (ColumnDef ColumnPosition)` syntax
- Optimize the queries whose `where` conditions are contradictory
- Optimize the `Add Index` operation to rectify the progress and reduce repetitive operations
- Optimize the `Index Look Join` operator to accelerate the query speed for small data size
- Fix the issue with prefix index judgment

9.5.7.2 Placement Driver (PD)

- Improve the stability of scheduling under exceptional situations

9.5.7.3 TiKV

- Support splitting table to ensure one region does not contain data from multiple tables
- Limit the length of a key to be no more than 4 KB
- More accurate read traffic statistics
- Implement deep protection on the coprocessor stack
- Fix the `LIKE` behavior and the `do_div_mod` bug

9.5.8 TiDB 1.0.1 Release Notes

On November 1, 2017, TiDB 1.0.1 is released with the following updates:

9.5.8.1 TiDB

- Support canceling DDL Job.
- Optimize the `IN` expression.
- Correct the result type of the `Show` statement.
- Support log slow query into a separate log file.
- Fix bugs.

9.5.8.2 TiKV

- Support flow control with write bytes.
- Reduce Raft allocation.
- Increase coprocessor stack size to 10MB.
- Remove the useless log from the coprocessor.

9.5.9 TiDB 1.0 Release Notes

On October 16, 2017, TiDB 1.0 is now released! This release is focused on MySQL compatibility, SQL optimization, stability, and performance.

9.5.9.1 TiDB

- The SQL query optimizer:
  - Adjust the cost model
  - Analyze pushdown
  - Function signature pushdown
- Optimize the internal data format to reduce the interim data size
- Enhance the MySQL compatibility
- Support the `NO_SQL_CACHE` syntax and limit the cache usage in the storage engine
- Refactor the Hash Aggregator operator to reduce the memory usage
- Support the Stream Aggregator operator

9.5.9.2 PD

- Support read flow based balancing
- Support setting the Store weight and weight based balancing

9.5.9.3 TiKV

- Coprocessor now supports more pushdown functions
- Support pushing down the sampling operation
- Support manually triggering data compact to collect space quickly
- Improve the performance and stability
- Add a Debug API for debugging
- TiSpark Beta Release:
- Support configuration framework
- Support ThriftSever/JDBC and Spark SQL

9.5.9.4 Acknowledgement

9.5.9.4.1 Special thanks to the following enterprises and teams

- Archon
- Mobike
- Samsung Electronics
- SpeedyCloud
- Tencent Cloud
- UCloud

9.5.9.4.2 Thanks to the open source software and services from the following organizations and individuals

- Asta Xie
- CNCF
- CoreOS
- Databricks
- Docker
- Github
- Grafana
- gRPC
- Jepsen
- Kubernetes
- Namazu
- Prometheus
- RedHat
- RocksDB Team
- Rust Team

9.5.9.4.3 Thanks to the individual contributors

- 8cbx
- Akihiro Suda
- aliyx
- alston111111
- andelf
- Andy Librian
- Arthur Yang
- astaxie
- Bai, Yang
- bailaohe
- Bin Liu
- Blame cosmos
- Breezewish
- Carlos Ferreira
- Ce Gao
- Changjian Zhang
- Cheng Lian
- Cholerae Hu
- Chu Chao
- coldwater
- Cole R Lawrence
- cuiqiu
- cuiyuan
- Cwen
- Dagang
- David Chen
- David Ding
- dawxy
- dcadevil
- Deshi Xiao
- Di Tang
- disksing
- dongxu
- dreamquster
- Drogon
- Du Chuan
- Dylan Wen
- eBoyy
- Eric Romano
- Ewan Chou
- Fiisio
- follitude
- Fred Wang

- fud
- fudali
- gaoyangxiaozhu
- Gogs
- goroutine
- Gregory Ian
- Guanqun Lu
- Guilherme Hübner Franco
- Haibin Xie
- Han Fei
- hawkingrei
- Hiroaki Nakamura
- hiwjd
- Hongyuan Wang
- Hu Ming
- Hu Ziming
- Huachao Huang
- HuaiyuXu
- Huxley Hu
- iamxy
- Ian
- insion
- iroi44
- Ivan.Yang
- Jack Yu
- jacky liu
- Jan Mercl
- Jason W
- Jay
- Jay Lee
- Jianfei Wang
- Jiaxing Liang
- Jie Zhou
- jinhelin
- Jonathan Boulle
- Karl Ostendorf
- knarfeh
- Kuiba
- leixuechun
- li
- Li Shihai
- Liao Qiang
- Light
- lijian
- Lilian Lee

- Liqueur Librazy
- Liu Cong
- Liu Shaohui
- liubo0127
- liyanan
- lkk2003rty
- Louis
- louishust
- luckcolors
- Lynn
- Mae Huang
- maiyang
- maxwell
- mengshangqi
- Michael Belenchenko
- mo2zie
- morefreeze
- MQ
- mxlxm
- Neil Shen
- netroby
- ngaut
- Nicole Nie
- nolouch
- onlymellb
- overvenus
- PaladinTyrion
- paulg
- Priya Seth
- qgxiaozhan
- qhsong
- Qiannan
- qiukeren
- qiuyesuifeng
- queenypingcap
- qupeng
- Rain Li
- ranxiaolong
- Ray
- Rick Yu
- shady
- ShawnLi
- Shen Li
- Sheng Tang
- Shirly

- Shuai Li
- ShuNing
- ShuYu Wang
- siddontang
- silenceper
- Simon J Mudd
- Simon Xia
- skimmilk6877
- sllt
- soup
- Sphinx
- Steffen
- sumBug
- sunhao2017
- Tao Meng
- Tao Zhou
- tennix
- tiancaiamao
- TianGuangyu
- Tristan Su
- ueizhou
- UncP
- Unknwon
- v01dstar
- Van
- WangXiangUSTC
- wangyanjun
- wangyisong1996
- weekface
- wegel
- Wei Fu
- Wenbin Xiao
- Wenting Li
- Wenxuan Shi
- winkyao
- woodpenker
- wuxuelian
- Xiang Li
- xiaojian cai
- Xuanjia Yang
- Xuanwo
- XuHuaiyu
- Yang Zhexuan
- Yann Autissier
- Yanzhe Chen

- Yiding Cui
- Yim
- youyouhu
- Yu Jun
- Yuwen Shen
- Zejun Li
- Zhang Yuning
- zhangjinpeng1987
- ZHAO Yijun
- Zhe-xuan Yang
- ZhengQian
- ZhengQianFang
- zhengwanbo
- ZhiFeng Hu
- Zhiyuan Zheng
- Zhou Tao
- Zhoubirdblue
- zhouningnan
- Ziyi Yan
- zs634134578
- zxylvlp
- zyguan
- zz-jason

9.5.10 Pre-GA Release Notes

On August 30, 2017, TiDB Pre-GA is released! This release is focused on MySQL compatibility, SQL optimization, stability, and performance.

9.5.10.1 TiDB

- The SQL query optimizer:
  - Adjust the cost model
  - Use index scan to handle the `where` clause with the `compare` expression which has different types on each side
  - Support the Greedy algorithm based Join Reorder
- Many enhancements have been introduced to be more compatible with MySQL
- Support `Natural Join`
- Support the JSON type (Experimental), including the query, update and index of the JSON fields
- Prune the useless data to reduce the consumption of the executor memory
- Support configuring prioritization in the SQL statements and automatically set the prioritization for some of the statements according to the query type
- Completed the expression refactor and the speed is increased by about 30%

9.5.10.2 Placement Driver (PD)

- Support manually changing the leader of the PD cluster

9.5.10.3 TiKV

- Use dedicated Rocksdb instance to store Raft log
- Use `DeleteRange` to speed up the deleting of replicas
- Coprocessor now supports more pushdown operators
- Improve the performance and stability

9.5.10.4 TiDB Connector for Spark Beta Release

- Implement the predicates pushdown
- Implement the aggregation pushdown
- Implement range pruning
- Capable of running full set of TPC+H except for one query that needs view support

9.5.11 TiDB RC4 Release Notes

On August 4, 2017, TiDB RC4 is released! This release is focused on MySQL compatibility, SQL optimization, stability, and performance.

9.5.11.1 Highlight

- For performance, the write performance is improved significantly, and the computing task scheduling supports prioritizing to avoid the impact of OLAP on OLTP.
- The optimizer is revised for a more accurate query cost estimating and for an automatic choice of the `Join` physical operator based on the cost.
- Many enhancements have been introduced to be more compatible with MySQL.
- TiSpark is now released to better support the OLAP business scenarios. You can now use Spark to access the data in TiKV.

9.5.11.2 Detailed updates

9.5.11.2.1 TiDB

- The SQL query optimizer refactoring:
  - Better support for TopN queries
  - Support the automatic choice of the of the `Join` physical operator based on the cost

  - Improved Projection Elimination

- The version check of schema is based on Table to avoid the impact of DDL on the ongoing transactions
- Support `BatchIndexJoin`
- Improve the `Explain` statement
- Improve the `Index Scan` performance
- Many enhancements have been introduced to be more compatible with MySQL
- Support the JSON type and operations
- Support the configuration of query prioritizing and isolation level

9.5.11.2.2 Placement Driver (PD)

- Support using PD to set the TiKV location labels
- Optimize the scheduler
  - PD is now supported to initialize the scheduling commands to TiKV.
  - Accelerate the response speed of the region heartbeat.
  - Optimize the `balance` algorithm
- Optimize data loading to speed up failover

9.5.11.2.3 TiKV

- Support the configuration of query prioritizing
- Support the RC isolation level
- Improve Jepsen test results and the stability
- Support Document Store
- Coprocessor now supports more pushdown functions
- Improve the performance and stability

9.5.11.2.4 TiSpark Beta Release

- Implement the prediction pushdown
- Implement the aggregation pushdown
- Implement range pruning
- Capable of running full set of TPC-H except one query that needs view support

9.5.12 TiDB RC3 Release Notes

On June 20, 2017, TiDB RC4 is released!This release is focused on MySQL compatibility, SQL optimization, stability, and performance.

9.5.12.1 Highlight

- The privilege management is refined to enable users to manage the data access privileges using the same way as in MySQL.
- DDL is accelerated.
- The load balancing policy and process are optimized for performance.
- TiDB Ansible is open sourced. By using TiDB-Ansilbe, you can deploy, upgrade, start and shutdown a TiDB cluster with one click.

9.5.12.2 Detailed updates

9.5.12.3 TiDB

- The following features are added or improved in the SQL query optimizer:
  - Support incremental statistics
  - Support the `Merge Sort Join` operator
  - Support the `Index Lookup Join` operator
  - Support the `Optimizer Hint` Syntax
  - Optimize the memory consumption of the `Scan`, `Join`, `Aggregation` operators
  - Optimize the Cost Based Optimizer (CBO) framework
  - Refactor `Expression`
- Support more complete privilege management
- DDL acceleration
- Support using HTTP API to get the data distribution information of tables
- Support using system variables to control the query concurrency
- Add more MySQL built-in functions
- Support using system variables to automatically split a big transaction into smaller ones to commit

9.5.12.4 Placement Driver (PD)

- Support gRPC
- Provide the Disaster Recovery Toolkit
- Use Garbage Collection to clear stale data automatically
- Support more efficient data balance
- Support hot Region scheduling to enable load balancing and speed up the data importing
- Performance
  - Accelerate getting Client TSO
  - Improve the efficiency of Region Heartbeat processing
- Improve the `pd-ctl` function

  - Update the Replica configuration dynamically
  - Get the Timestamp Oracle (TSO)
  - Use ID to get the Region information

9.5.12.5 TiKV

- Support gRPC
- Support the Sorted String Table (SST) format snapshot to improve the load balancing speed of a cluster
- Support using the Heap Profile to uncover memory leaks
- Support Streaming SIMD Extensions (SSE) and speed up the CRC32 calculation
- Accelerate transferring leader for faster load balancing
- Use Batch Apply to reduce CPU usage and improve the write performance
- Support parallel Prewrite to improve the transaction write speed
- Optimize the scheduling of the coprocessor thread pool to reduce the impact of big queries on point get
- The new Loader supports data importing at the table level, as well as splitting a big table into smaller logical blocks to import concurrently to improve the data importing speed.

9.5.13 TiDB RC2 Release Notes

On August 4, 2017, TiDB RC4 is released! This release is focused on the compatibility with MySQL, SQL query optimizer, system stability and performance in this version. What's more, a new permission management mechanism is added and users can control data access in the same way as the MySQL privilege management system.

9.5.13.1 TiDB

- Query optimizer
  - Collect column/index statistics and use them in the query optimizer
  - Optimize the correlated subquery
  - Optimize the Cost Based Optimizer (CBO) framework
  - Eliminate aggregation using unique key information
  - Refactor expression evaluation framework
  - Convert Distinct to GroupBy
  - Support the topn operation push-down
- Support basic privilege management
- Add lots of MySQL built-in functions
- Improve the Alter Table statement and support the modification of table name, default value and comment
- Support the Create Table Like statement

- Support the Show Warnings statement
- Support the Rename Table statement
- Restrict the size of a single transaction to avoid the cluster blocking of large transactions
- Automatically split data in the process of Load Data
- Optimize the performance of the AddIndex and Delete statement
- Support "ANSI_QUOTES" sql_mode
- Improve the monitoring system
- Fix Bugs
- Solve the problem of memory leak

9.5.13.2 PD

- Support location aware replica scheduling
- Conduct fast scheduling based on the number of region
- pd-ctl support more features
  - Add or delete PD
  - Obtain Region information with Key
  - Add or delete scheduler and operator
  - Obtain cluster label information

9.5.13.3 TiKV

- Support Async Apply to improve the entire write performance
- Use prefix seek to improve the read performance of Write CF
- Use memory hint prefix to improve the insert performance of Raft CF
- Optimize the single read transaction performance
- Support more push-down expressions
- Improve the monitoring system
- Fix Bugs

9.5.14 TiDB RC1 Release Notes

On December 23, 2016, TiDB RC1 is released. See the following updates in this release:

9.5.14.1 TiKV

- The write speed has been improved.
- The disk space usage is reduced.
- Hundreds of TBs of data can be supported.
- The stability is improved and TiKV can support a cluster with 200 nodes.
- Supports the Raw KV API and the Golang client.

9.5.14.2 Placement Driver (PD)

- The scheduling strategy framework is optimized and now the strategy is more flexible and reasonable.
- The support for `label` is added to support Cross Data Center scheduling.
- PD Controller is provided to operate the PD cluster more easily.

9.5.14.3 TiDB

- The following features are added or improved in the SQL query optimizer:
  - Eager aggregation
  - More detailed `EXPLAIN` information
  - Parallelization of the `UNION` operator
  - Optimization of the subquery performance
  - Optimization of the conditional push-down
  - Optimization of the Cost Based Optimizer (CBO) framework
- The implementation of the time related data types are refactored to improve the compatibility with MySQL.
- More built-in functions in MySQL are supported.
- The speed of the `add index` statement is enhanced.
- The following statements are supported:
  - Use the `CHANGE COLUMN` statement to change the name of a column.
  - Use `MODIFY COLUMN` and `CHANGE COLUMN` of the `ALTER TABLE` statement for some of the column type transfer.

9.5.14.4 New tools

- `Loader` is added to be compatible with the `mydumper` data format in Percona and provides the following functions:
  - Multi-thread import
  - Retry if error occurs
  - Breakpoint resume
  - Targeted optimization for TiDB
- The tool for one-click deployment is added.

10 Glossary

10.1 A

10.1.1 ACID

ACID refers to the four key properties of a transaction: atomicity, consistency, isolation, and durability. Each of these properties is described below.

- **Atomicity** means that either all the changes of an operation are performed, or none of them are. TiDB ensures the atomicity of the Region that stores the Primary Key to achieve the atomicity of transactions.
- **Consistency** means that transactions always bring the database from one consistent state to another. In TiDB, data consistency is ensured before writing data to the memory.
- **Isolation** means that a transaction in process is invisible to other transactions until it completes. This allows concurrent transactions to read and write data without sacrificing consistency. TiDB currently supports the isolation level of `REPEATABLE READ`.
- **Durability** means that once a transaction is committed, it remains committed even in the event of a system failure. TiKV uses persistent storage to ensure durability.

10.2 L

10.2.1 leader/follower/learner

Leader/Follower/Learner each corresponds to a role in a Raft group of peers. The leader services all client requests and replicates data to the followers. If the group leader fails, one of the followers will be elected as the new leader. Learners are non-voting followers that only serves in the process of replica addition.

10.3 O

10.3.1 Operator

An operator is a collection of actions that applies to a Region for scheduling purposes. Operators perform scheduling tasks such as "migrate the leader of Region 2 to Store 5" and "migrate replicas of Region 2 to Store 1, 4, 5".

An operator can be computed and generated by a scheduler, or created by an external API.

10.3.2 Operator step

An operator step is a step in the execution of an operator. An operator normally contains multiple Operator steps.

Currently, available steps generated by PD include:

- `TransferLeader`: Transfers leadership to a specified member
- `AddPeer`: Adds peers to a specified store
- `RemovePeer`: Removes a peer of a Region
- `AddLearner`: Adds learners to a specified store
- `PromoteLearner`: Promotes a specified learner to a voting member
- `SplitRegion`: Splits a specified Region into two

10.4 P

10.4.1 pending/down

"Pending" and "down" are two special states of a peer. Pending indicates that the Raft log of followers or learners is vastly different from that of leader. Followers in pending cannot be elected as leader. "Down" refers to a state that a peer ceases to respond to leader for a long time, which usually means the corresponding node is down or isolated from the network.

10.5 R

10.5.1 Region/peer/Raft group

Region is the minimal piece of data storage in TiKV, each representing a range of data (96 MiB by default). Each Region has three replicas by default. A replica of a Region is called a peer. Multiple peers of the same Region replicate data via the Raft consensus algorithm, so peers are also members of a Raft instance. TiKV uses Multi-Raft to manage data. That is, for each Region, there is a corresponding, isolated Raft group.

10.5.2 Region split

Regions are generated as data writes increase. The process of splitting is called Region split.

The mechanism of Region split is to use one initial Region to cover the entire key space, and generate new Regions through splitting existing ones every time the size of the Region or the number of keys has reached a threshold.

10.5.3 restore

Restore is the reverse of the backup operation. It is the process of bringing back the system to an earlier state by retrieving data from a prepared backup.

10.6 S

10.6.1 scheduler

Schedulers are components in PD that generate scheduling tasks. Each scheduler in PD runs independently and serves different purposes. The commonly used schedulers are:

- `balance-leader-scheduler`: Balances the distribution of leaders
- `balance-region-scheduler`: Balances the distribution of peers
- `hot-region-scheduler`: Balances the distribution of hot Regions
- `evict-leader-{store-id}`: Evicts all leaders of a node (often used for rolling upgrades)

10.6.2 Store

A store refers to the storage node in the TiKV cluster (an instance of `tikv-server`). Each store has a corresponding TiKV instance.
