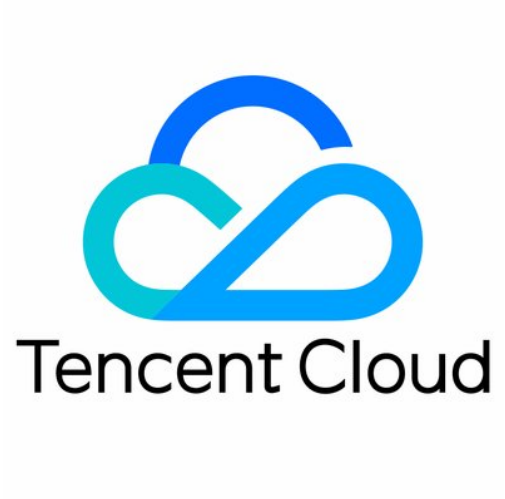
TDSQL for MySQL

Best Practice

Product Documentation



Contents

Best Practice
Import from Standalone Instance to TDSQL Instance

Last updated：2024-01-06 17:34:55

As the distributed architecture of a distributed database is imperceptible to the user, in general, only the table structure needs to be created in advance, and you can then migrate data using a MySQL client such as mysqldump, Navicat, or SQLyog in the following steps:

Step 1. Prepare the import and export environments

Step 2. Export the table structure and data of the source table

Step 3. Modify the statement for table creation and create a table structure in the target table

Step 4. Import data

Step 1. Prepare the import and export environments

Before migrating data, you need to get the following prepared:

A CVM instance

The specification of 2 CPU cores, 8 GB memory, and 500+ GB disk capacity (depending on data volume) is recommended

Linux

MySQL client installed

If the data volume is small (less than 10 GB), you can also import it directly over the public network (internet) with nothing prepared

TDSQL

Select a capacity as expected and perform initialization according to the source database character set, table name case sensitivity, and value of `innodb_page_size` .

Create an account with all global permissions enabled (recommended)

Enable a public IP address if necessary

Step 2. Export the table structure and data of the source table

Demonstration environment

Manipulated database: caccts

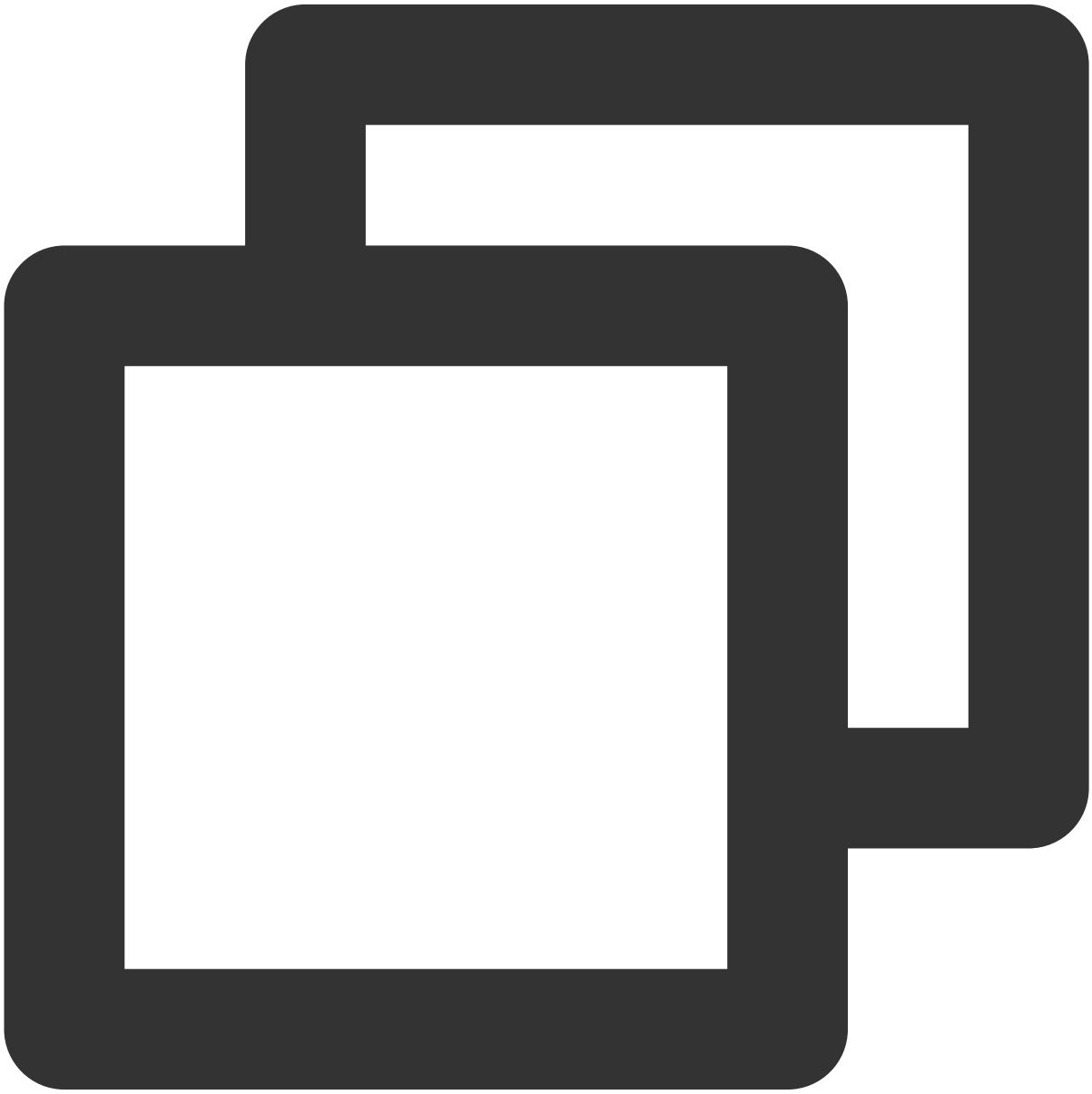
Manipulated table: t_acct_water_0

Source database: single-instance MySQL

Target database: TDSQL for Percona or MariaDB

Exporting table structure from source database

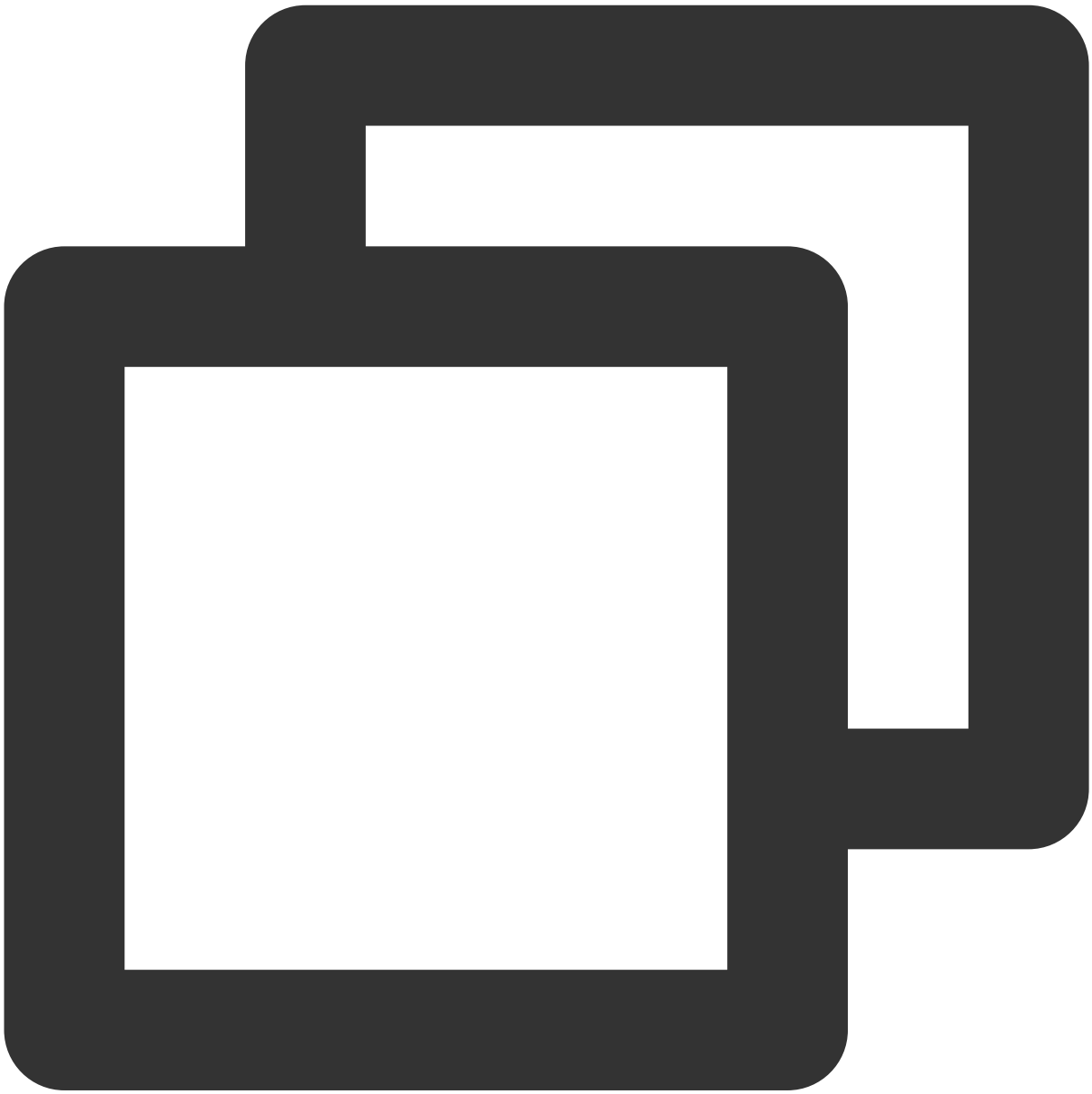
Run `mysqldump -u username -p password -d dbname tablename > tablename.sql` to export the table structure



```
// Command instance
mysqldump -utest -ptest1234 -d -S /data/4003/prod/mysql.sock caccts t_acct_water_0
```

Exporting table data from source database

Run `mysqldump -c -u username -p password dbname tablename > tablename.sql` to export the table data.



```
// Command instance
mysqldump -c -t -utest -ptest1234  -S /data/4003/prod/mysql.sock caccts t_acct_wate
```

Note:

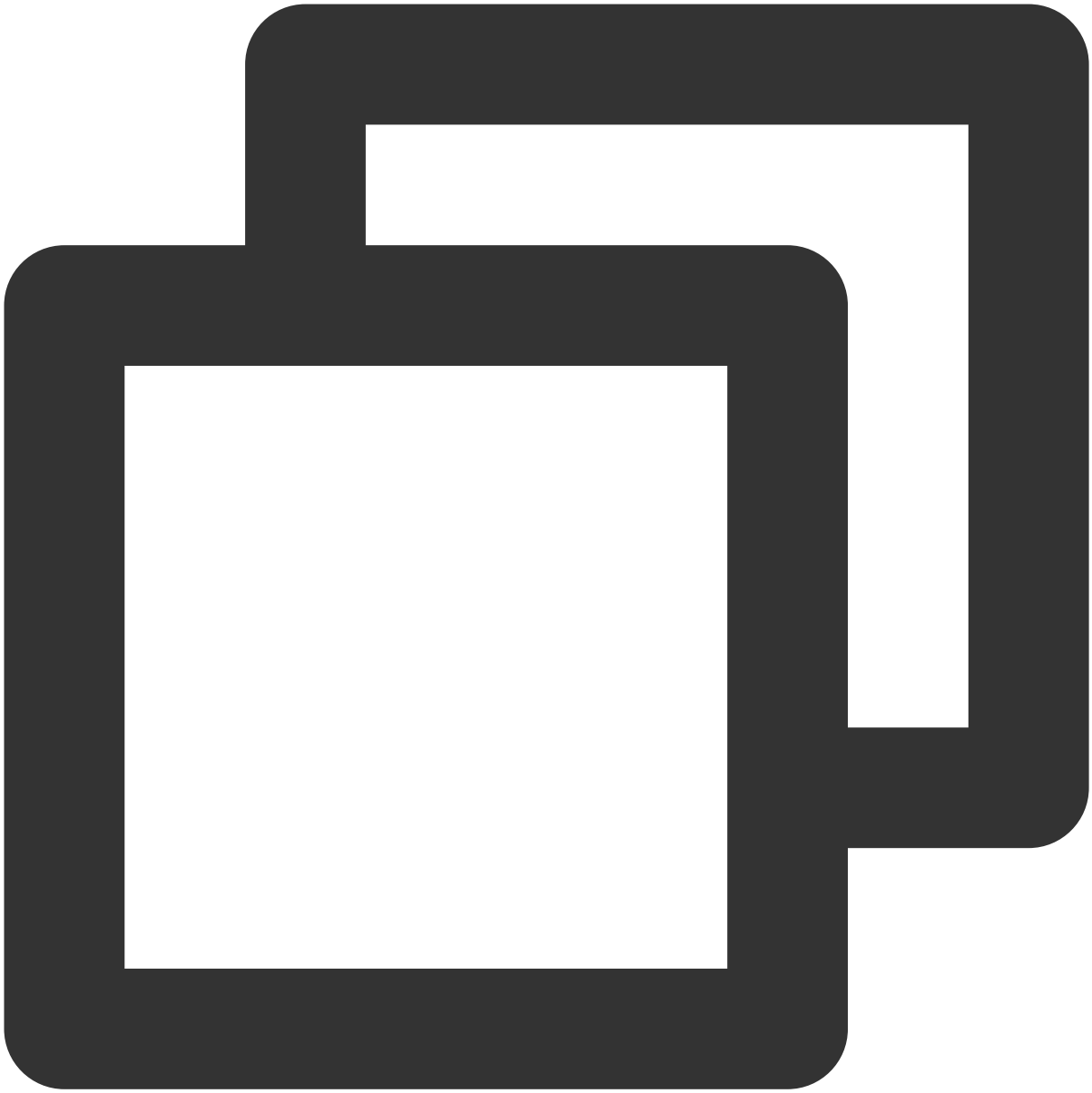
Data must be exported using mysqldump with the `-c` parameter appended, because only in this way can all the exported data rows have a column name field. SQL data rows with no column name field will be rejected by TDSQL for Percona or MariaDB. `-t` indicates to export the table data only but not the table structure.

Uploading files to directory on CVM instance

Before uploading files, you need to enable the public IP address of the CVM instance. Then, upload at least the following files just exported as instructed in Uploading Files to Linux CVM Through SCP:
Table structure SQL: table.sql
Data SQL: data.sql

Step 3. Modify the statement for table creation and create a table structure in the target table

Open the exported table structure file `table.sql`, add the primary key and shardkey through statements like the following ones, and save it as `tablenew.sql`.



```
CREATE TABLE (
column name 1 data type,
column name 2 data type,
column name 3 data type,
....,
PRIMARY KEY('column name n'))
ENGINE=INNODB DEFAULT CHARSET=xxxx
shardkey=keyname
```

Note:

Primary key and shardkey must be set. Pay attention to the table name case sensitivity. You are recommended to delete redundant comments; otherwise, the table may not be created successfully.

```
CREATE TABLE `t_acct_water_0` (
  `Fseq_no` int(11) NOT NULL DEFAULT '0',
  `Facct_id` varchar(64) NOT NULL DEFAULT '',
  `Facct_key` varchar(128) NOT NULL DEFAULT '',
  `Ffirstkey` varchar(64) NOT NULL DEFAULT '',
  `Fuin` varchar(64) NOT NULL DEFAULT '',
  `Fuserid` varchar(64) NOT NULL DEFAULT '',
  `Factionid` varchar(32) NOT NULL DEFAULT '',
  `Fzoneid` varchar(32) NOT NULL DEFAULT '',
  `Froleid` varchar(32) NOT NULL DEFAULT '',
  `Fsubacctid` varchar(64) NOT NULL DEFAULT '',
  `Fextern_tran_type` int(11) NOT NULL DEFAULT '0',
  `Fbase_tran_type` int(11) NOT NULL DEFAULT '0',
  `Fwater_type` int(11) NOT NULL DEFAULT '0',
  `Fio_flag` int(11) NOT NULL DEFAULT '0',
  `Fbill_no` varchar(64) NOT NULL DEFAULT '',
  `Fportal_seq` varchar(256) NOT NULL DEFAULT '',
  `Ftran_amt` bigint(20) NOT NULL DEFAULT '0',
  `Fbalance` bigint(20) NOT NULL DEFAULT '0',
  `Fgen_tran_amt` bigint(20) NOT NULL DEFAULT '0',
  `Fgen_balance` bigint(20) NOT NULL DEFAULT '0',
  `Fcreate_time` int(11) NOT NULL DEFAULT '0',
  `Fsource` varchar(32) NOT NULL DEFAULT '',
  `Fdevice` varchar(32) NOT NULL DEFAULT '',
  `Fcheck_account` varchar(256) NOT NULL DEFAULT '',
  `Fclient_ip` varchar(32) NOT NULL DEFAULT '',
  `Fuser_ip` varchar(32) NOT NULL DEFAULT '',
  `Ftran_info` varchar(256) NOT NULL DEFAULT '',
  `Fremark` varchar(256) NOT NULL DEFAULT '',
  `Freserve1` varchar(256) NOT NULL DEFAULT '',
  `Freserve2` varchar(256) NOT NULL DEFAULT
  `blobtest` blob,
  PRIMARY KEY (`Fseq_no`),
  KEY `Ffirstkey` (`Ffirstkey`),
  KEY `s_acctid_acctkey` (`Facct_id`,`Facct_key`),
  KEY `i_create_time` (`Fcreate_time`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 shardkey=Fseq_no
```

Step 4. Import data

Connecting to TDSQL for Percona or MariaDB instance

On the CVM instance, run `mysql -u username -p password -h IP -P port` to log in to the MySQL server, and run `use dbname` to enter the database.

Note:

You may need to create a database first.

Importing table structure

Use the uploaded files to import data by running the `source` command.

1. First, import the table structure: `source /file path/tablenew.sql`
2. Then, import the data: `source /file path/data.sql`
3. Verify the import result: `select  count(*) from tablename`

Note:

You need to import the table creation statements first and then import the data. You can also import the .sql files directly through MySQL's `source` command.

Other Solutions

Generally, you can import data smoothly as long as you have created the corresponding table structure with the shardkey specified in the target table before importing data.

Import Between TDSQL Instances

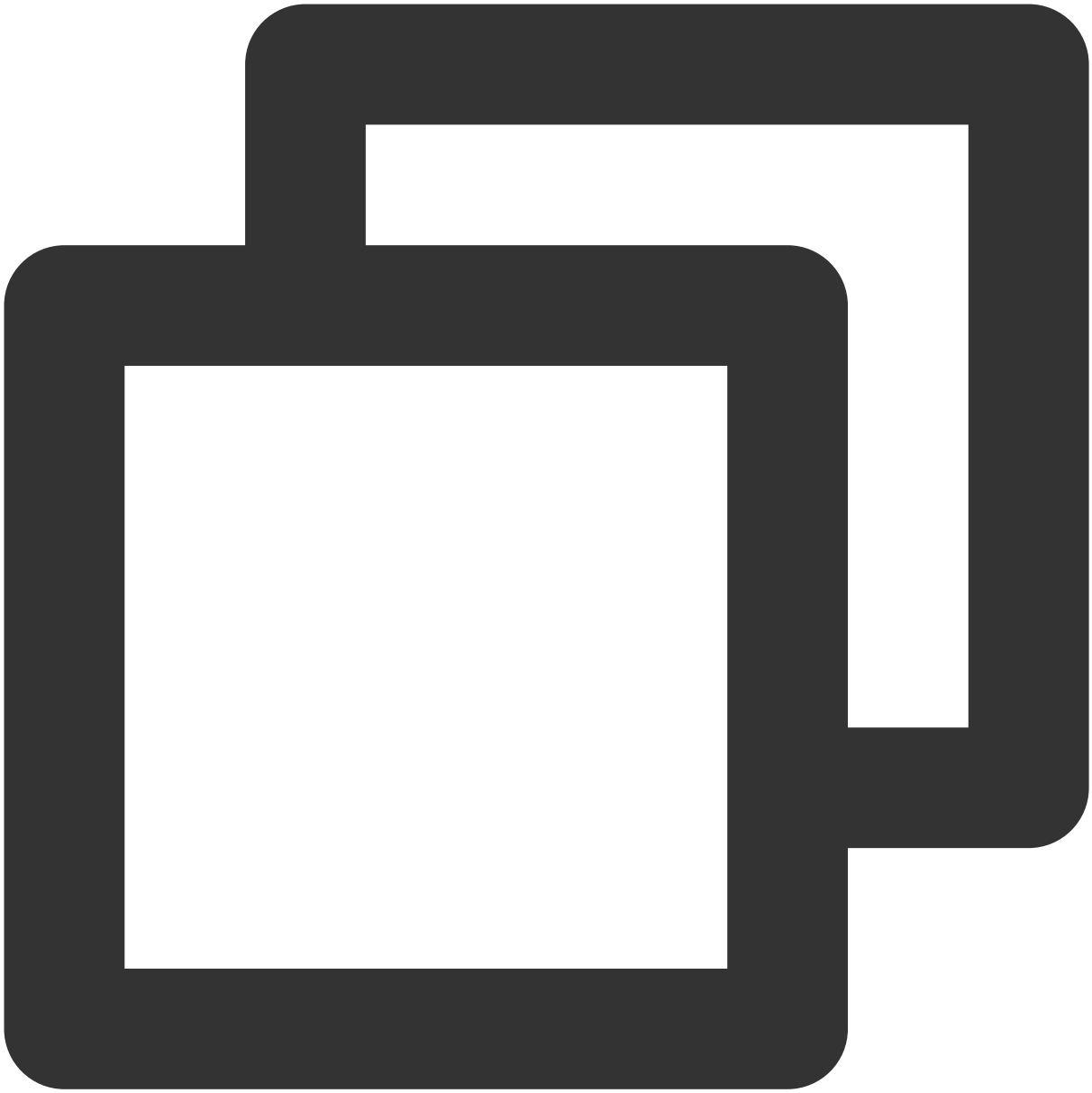
Last updated：2024-01-06 17:34:55

The solution for data importing from one distributed database to another is special. mysqldump is used as an example below to summarize the importing steps:

1. Install mysqldump (for MariaDB)

Purchase a Linux CVM instance and run `yum install mariadb-server` to install mysqldump.

2. Export the table structure



```
mysqldump --compact --single-transaction -d -uxxx -pxxx  -hxxx.xxx.xxx.xxx -Pxxxx
```

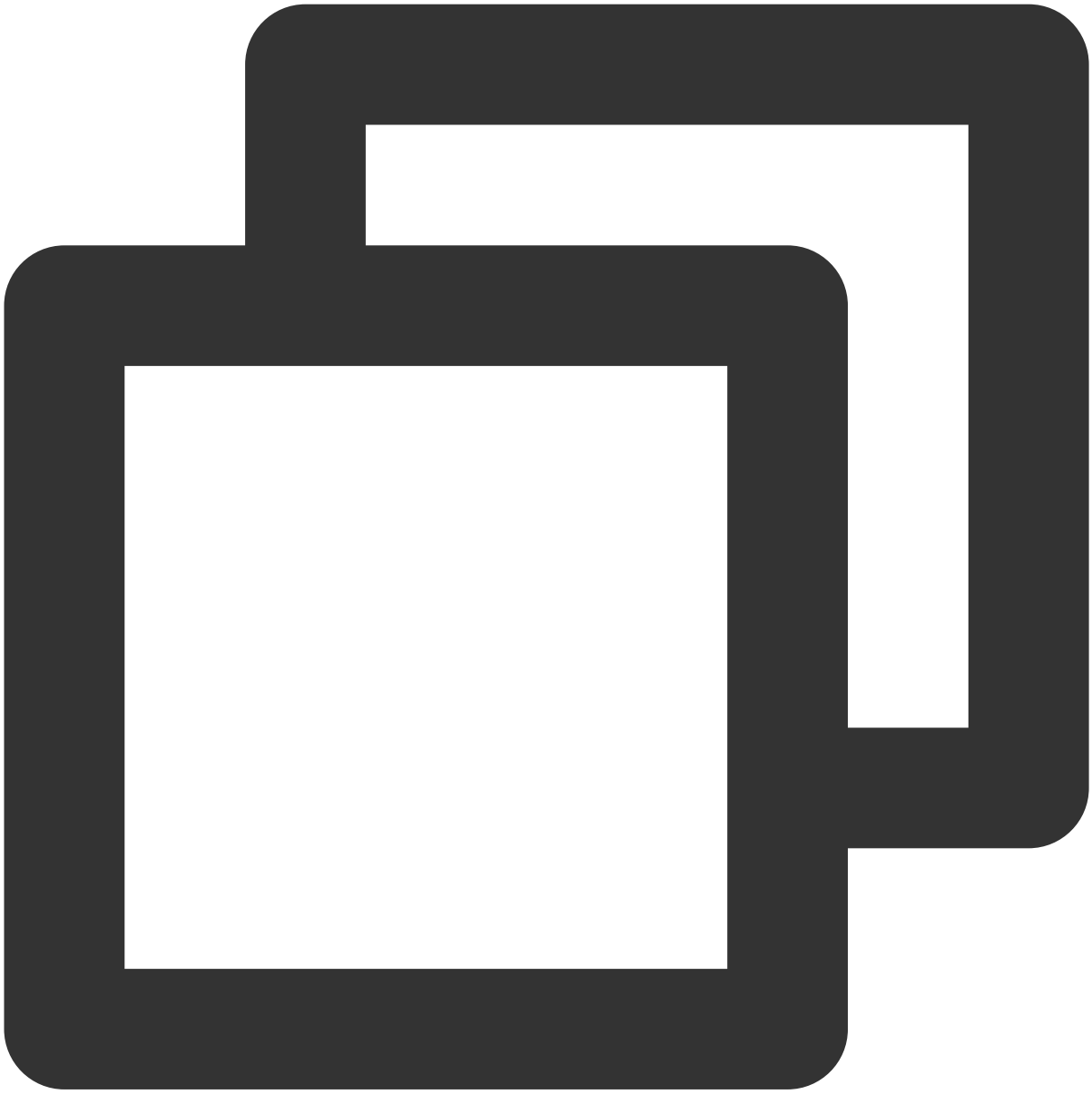
Note:

Please select the `db_name` and `table_name` parameters as needed.

3. Export data

Export data by using mysqldump:

Set the `net_write_timeout` parameter in the parameter settings in the TDSQL for MySQL Console: set global net_write_timeout=28800



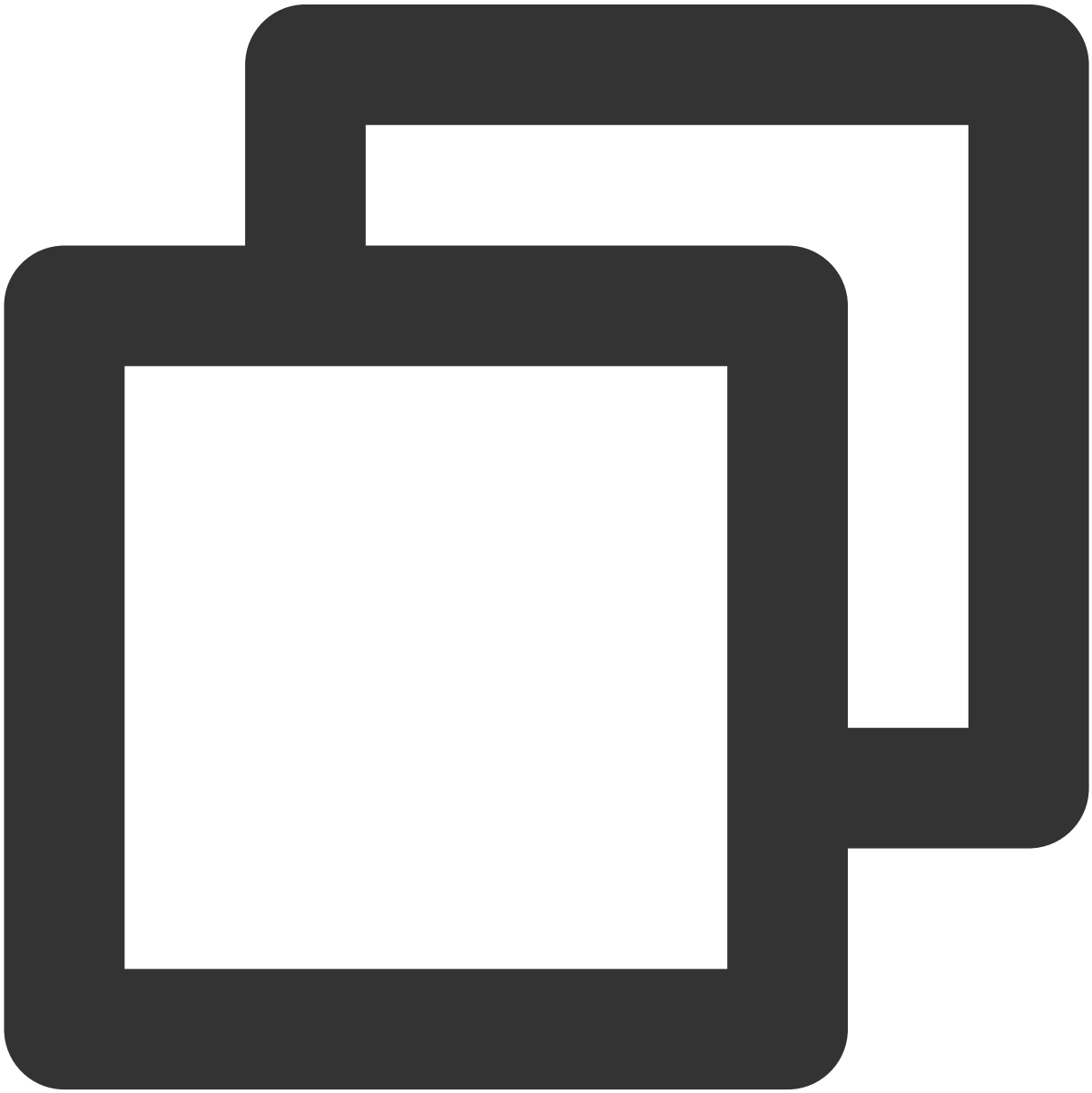
```
mysqldump --compact --single-transaction --no-create-info -c -uxxx  -pxxx -hxxx.xxx
```

Note:

The `db_name` and `table_name` parameters should be selected as needed. If the exported data is to be imported into another set of TDSQL for MySQL environment, the `-c` option must be added, and there should be a space between `-c` and `db_name` .

```
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]# mysqldump --compact --single-transaction -d -u -p -h colin_test test1 > schema.sql
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]# mysqldump --compact --single-transaction --no-create-info -c -u -p -h colin_test test1 > data.sql
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]#
```

4. Create a database in the target instance



```
mysql --default-character-set=utf8 -uxxx -pxxx -hxxx.xxx.xxx.xxx -Pxxxx -e "create
```

--default-character-set=utf8: set based on your target table.

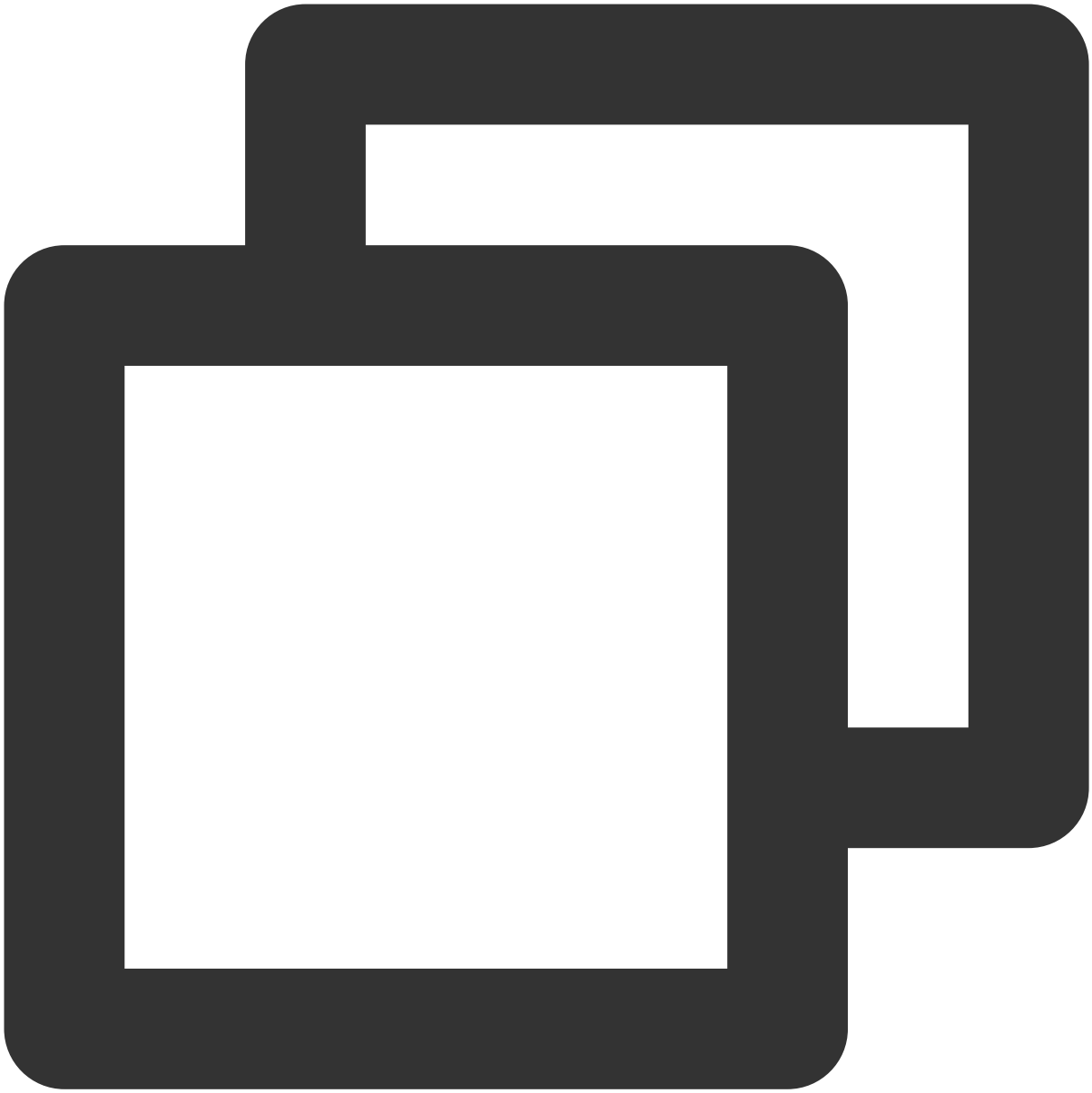
-uxxx: a privileged account (`-u` is a keyword).

-pxxx: password (`-p` is a keyword).

-hxxx.xxx.xxx.xxx -Pxxxx: IP and port of the database instance.

dbname: database name.

5. Import the table structure into the target instance



```
mysql --default-character-set=utf8 -uxxx -pxxx -hxxx.xxx.xxx.xxx -Pxxxx dbname < sc
```

--default-character-set=utf8: set based on your target table.

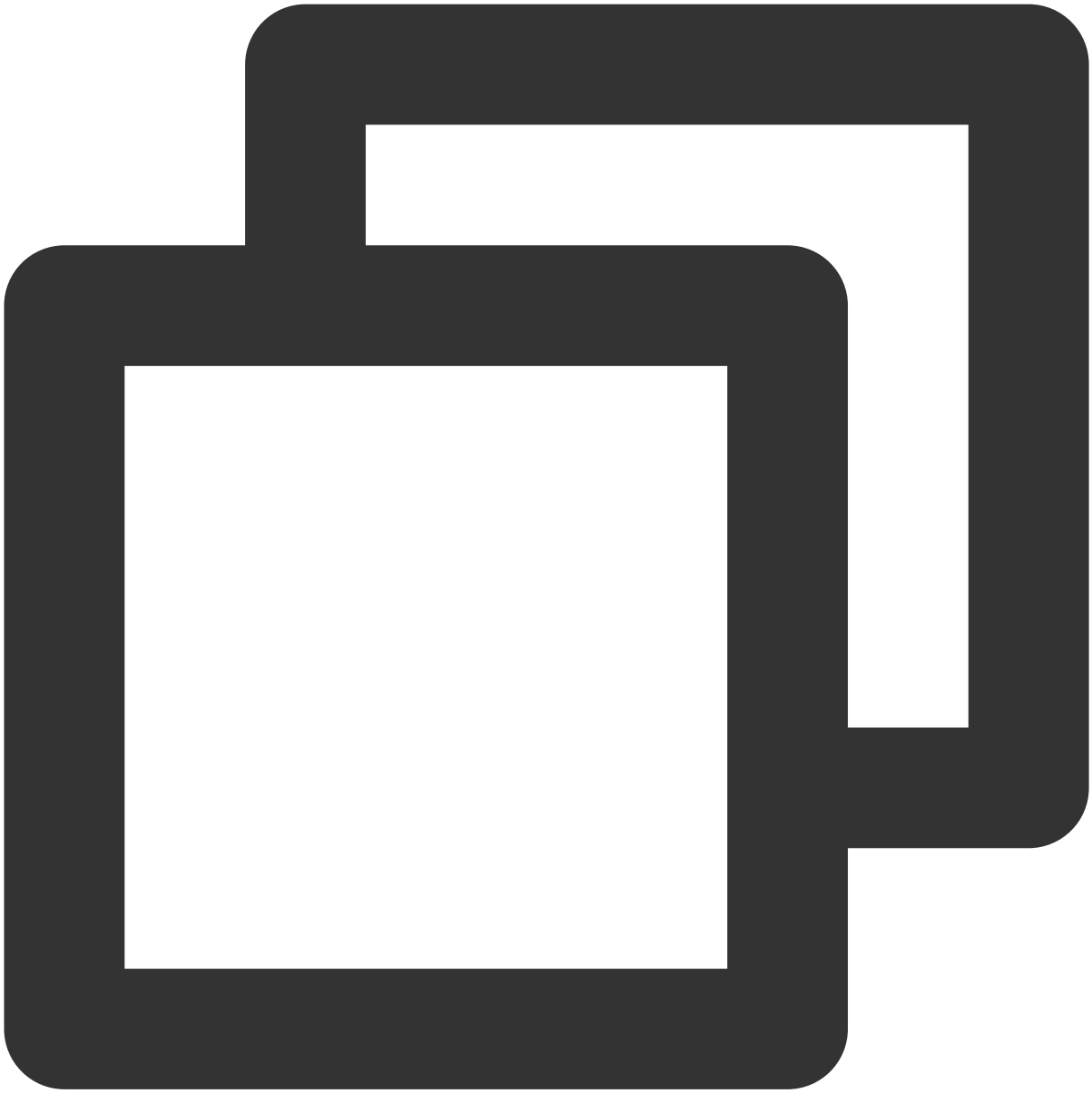
-uxxx: a privileged account (`-u` is a keyword).

-pxxx: password (`-p` is a keyword).

-hxxx.xxx.xxx.xxx -Pxxxx: IP and port of the database instance.

dbname: database name.

6. Import table data into the target instance



```
mysql --default-character-set=utf8 -uxxx -pxxx -hxxx.xxx.xxx.xxx -Pxxxx dbname < da
```

Note:

If an auto-increment field is used in the source table, and the "Column 'xx' specified twice" error occurs during import, the `schema.sql` needs to be processed.

Remove the back quotes from the auto-increment field (cat schema.sql | tr " `" " " > schema_tr.sql), drop database, and repeat steps 3-5 by using the processed` schema_tr.sql`.

```
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]# mysql --default-character-set=utf8 -u  -p  -h  -e "create database colin_test;"
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]# mysql --default-character-set=utf8 -u  -p  -h  colin_test <  schema.sql
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]#
[root@VM_32_32_centos ~]# mysql --default-character-set=utf8 -u  -p  -h  colin_test <  data.sql
[root@VM_32_32_centos ~]#
```

Selection of TDSQL Instance and Shard Configuration

Last updated：2024-01-06 17:34:55

TDSQL Selection Overview

A TDSQL instance is composed of shards. The specification and number of shards determine the processing capability of the instance. In theory:
TDSQL instance read and write concurrence performance = ∑ (performance of a shard with a certain specification * number of shards)
TDSQL instance transaction performance = ∑ (transaction performance of a shard with a certain specification * 70% * number of shards)
Therefore, the higher the shard specification and quantity, the stronger the processing capability of the instance. The performance of a shard is mainly subject to CPU/memory and measured by QPS. General performance metrics are detailed in the shard performance description section.

TDSQL Shard Specification Selection

Specification of a TDSQL shard is determined by three factors: performance, capacity, and other requirements.
Performance: by predicting the performance demand and possible growth for at least six months, you can define the total CPU/memory size required of your TDSQL instance.
Capacity: by predicting the disk capacity and possible growth for at least one year, you can define the total disk size required of your TDSQL instance.
Other requirements: you are recommended to store at least **50 million rows of data** in one shard and take into account business needs such as broadcast and non-sharded tables and in-node joins.
Note:
You are recommended to ensure high specification for a single shard while keeping the shard quantity relatively small. Based on the above, it is estimated that you may have the following business requirements. Recommended strategies are as follows:
For functional testing in TDSQL with no special performance requirements: 2 shards with **2 GB of memory and 25 GB of disk capacity each**.
In initial stage of business when the total size of data is small but grows fast: 2 shards with **16 GB of memory and 200 GB of disk capacity each**.

In stable development stage when sharding is based on actual business conditions: 4 shards, and the specification for each one should be **current business peak * growth rate / 4**.

TDSQL Shard Performance Test

The database benchmark performance test is to modify the descriptions of sysbench 0.5: the OTLP script that comes with sysbench was modified. Specifically, the read/write ratio was changed to 1:1 and controlled by the testing command parameters `oltp_point_selects` and `oltp_index_updates`. In this document, all test cases involve four SELECT operations and one UPDATE operation with the read/write ratio at 4:1.

Memory (GB)	Storage Capacity (GB)	Dataset (GB)	Number of Clients	Single-Client Concurrence	QPS	TPS
2 GB	100 GB	46 GB	1	128	1880	351
4 GB	200 GB	76 GB	1	128	3983	797
8 GB	200 GB	142 GB	1	128	6151	1210
16 GB	400 GB	238 GB	1	128	10098	2119
32 GB	700 GB	238 GB	2	128	20125	3549
64 GB	1 TB	378 GB	2	128	37956	7002
96 GB	1.5 TB	378 GB	3	128	51026	10591
120 GB	2 TB	378 GB	3	128	81050	15013
240 GB	3 TB	567 GB	4	128	96891	17698
480 GB	6 TB	567 GB	6	128	140256	26599

TPS here is for single transaction other than distributed transaction.
As required by operational strategies, current TDSQL instances (or part of them) adopt the policy of overuse of idle resources, so CPU utilization may exceed 100% on your monitoring panel.