

腾讯云分布式数据库 DCDB

最佳实践

产品文档



腾讯云

【版权声明】

©2013-2017 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

文档声明.....	2
最佳实践.....	4
DCDB开发指南.....	4
从单机实例导入到分布式实例	40
如何选择DCDB实例配置和分片配置.....	46
从分布式实例导入到分布式实例	48

最佳实践

DCDB开发指南

DCDB 开发指南

DCDB提供和mysql兼容的连接方式，用户通过IP地址、端口号以及用户名、密码连接DCDB系统：

```
mysql -h10.231.136.34 -P3306 -utest12 -ptest123 -c
```

注意：DCDB不支持4.0以下的版本以及压缩协议，建议在使用客户端的时候增加

-c

选项，以便于使用某些高级功能。

概述

DCDB实现水平分表，业务只需在创建表的时候指定一个分表字段，后续操作对业务透明，由DCDB负责数据的路由以及汇总。

- 提供了灵活的读写分离模式
- 支持全局的order by，group by，limit操作；
- 聚合函数支持sum，count，avg，min，max(其他聚合函数不支持)
- 支持单set的join，基于分组以及小表在一定程度上实现跨set的join；
- 支持预处理协议；
- 支持全局唯一字段；
- 提供特定的sql查询整个集群的配置和状态；
- 支持分布式事务
- 支持两级分区

不支持MySQL的一些特性：

- 自定义函数
- 视图、存储过程、触发器，游标
- 外键，自建分区
- 复合语句，如BEGIN END，LOOP，UNION
- 子查询，having字句，（但支持带shardkey的derived table）

语言结构

DCDB支持所有MySQL使用的文字格式，包括：

String Literals

Numeric Literals

Date and Time Literals

Hexadecimal Literals

Bit-Value Literals

Boolean Literals

NULL Values

String Literals

String Literals 是一个 bytes 或者 characters

的序列，两端被单引号'或者双引号"包围，目前DCDB不支持ANSI_QUOTES SQL

MODE，双引号"包围的始终认为是String Literals，而不是identifier

不支持 character set introducer，即[_charset_name]'string' [COLLATE collation_name]这种格式

支持的转义字符：

\0: ASCII NUL (X' 00') 字符

\ ': 单引号

\ ": 双引号

\b: 退格符号
\n: 换行符
\r: 回车符
\t: tab 符 (制表符)
\z: ASCII 26 (Ctrl + Z)
\: 反斜杠 \
\%: \%
\: _

Numeric Literals

数值字面值包括 integer 跟 Decimal 类型跟浮点数字面值。

integer 可以包括 . 作为小数点分隔，数字前可以有 - 或者 + 来表示正数或者负数。

精确数值字面值可以表示为如下格式：1, .2, 3.4, -5, -6.78, +9.10.

科学记数法，如下格式：1.2E3, 1.2E-3, -1.2E3, -1.2E-3。

Date and Time Literals

DATE支持如下格式：

'YYYY-MM-DD' or 'YY-MM-DD'

'YYYYMMDD' or 'YYMMDD'

YYYYMMDD or YYMMDD

如：'2012-12-31', '2012/12/31', '2012^12^31', '2012@12@31' '20070523', '070523'

DATETIME，TIMESTAMP支持如下格式：

'YYYY-MM-DD HH:MM:SS' or 'YY-MM-DD HH:MM:SS'

'YYYYMMDDHHMMSS' or 'YYMMDDHHMMSS'

YYYYMMDDHHMMSS or YYMMDDHHMMSS

如'2012-12-31 11:30:45', '2012^12^31 11+30+45', '2012/12/31 11*30*45', '2012@12@31
11^30^45', 19830905132800

Hexadecimal Literals

支持格式如下：

X'01AF'

X'01af'

x'01AF'

x'01af'

0x01AF

0x01af

Bit-Value Literals

支持格式如下：

b'01'

B'01'

0b01

Boolean Literals

常量 TRUE 和 FALSE 等于 1 和 0，它是大小写不敏感的。

```
mysql> SELECT TRUE, true, FALSE, false;
```

```
+-----+-----+-----+-----+
```

```
| TRUE | TRUE | FALSE | FALSE |
+-----+-----+-----+-----+
| 1 | 1 | 0 | 0 |
+-----+-----+-----+
1 row in set (0.03 sec)
```

NULL Values

NULL 代表数据为空，它是大小写不敏感的，与 \N(大小写敏感) 同义。

需要注意的是 NULL 跟 0 并不一样，跟空字符串 "" 也不一样。

字符集和时区

DCDB在后端存储支持MySQL的所有字符集和字符序

```
mysql> show character set;
+-----+-----+-----+-----+
| Charset | Description | Default collation | Maxlen |
+-----+-----+-----+-----+
| big5 | Big5 Traditional Chinese | big5_chinese_ci | 2 |
| dec8 | DEC West European | dec8_swedish_ci | 1 |
| cp850 | DOS West European | cp850_general_ci | 1 |
| hp8 | HP West European | hp8_english_ci | 1 |
| koi8r | KOI8-R Relcom Russian | koi8r_general_ci | 1 |
| latin1 | cp1252 West European | latin1_swedish_ci | 1 |
| latin2 | ISO 8859-2 Central European | latin2_general_ci | 1 |
| swe7 | 7bit Swedish | swe7_swedish_ci | 1 |
| ascii | US ASCII | ascii_general_ci | 1 |
| ujis | EUC-JP Japanese | ujis_japanese_ci | 3 |
| sjis | Shift-JIS Japanese | sjis_japanese_ci | 2 |
| hebrew | ISO 8859-8 Hebrew | hebrew_general_ci | 1 |
| tis620 | TIS620 Thai | tis620_thai_ci | 1 |
```



```
| euckr | EUC-KR Korean | euckr_korean_ci | 2 |
| koi8u | KOI8-U Ukrainian | koi8u_general_ci | 1 |
| gb2312 | GB2312 Simplified Chinese | gb2312_chinese_ci | 2 |
| greek | ISO 8859-7 Greek | greek_general_ci | 1 |
| cp1250 | Windows Central European | cp1250_general_ci | 1 |
| gbk | GBK Simplified Chinese | gbk_chinese_ci | 2 |
| latin5 | ISO 8859-9 Turkish | latin5_turkish_ci | 1 |
| armSCII8 | ARMSCII-8 Armenian | armSCII8_general_ci | 1 |
| utf8 | UTF-8 Unicode | utf8_general_ci | 3 |
| ucs2 | UCS-2 Unicode | ucs2_general_ci | 2 |
| cp866 | DOS Russian | cp866_general_ci | 1 |
| keybcs2 | DOS Kamenicky Czech-Slovak | keybcs2_general_ci | 1 |
| macce | Mac Central European | macce_general_ci | 1 |
| macroman | Mac West European | macroman_general_ci | 1 |
| cp852 | DOS Central European | cp852_general_ci | 1 |
| latin7 | ISO 8859-13 Baltic | latin7_general_ci | 1 |
| utf8mb4 | UTF-8 Unicode | utf8mb4_general_ci | 4 |
| cp1251 | Windows Cyrillic | cp1251_general_ci | 1 |
| utf16 | UTF-16 Unicode | utf16_general_ci | 4 |
| utf16le | UTF-16LE Unicode | utf16le_general_ci | 4 |
| cp1256 | Windows Arabic | cp1256_general_ci | 1 |
| cp1257 | Windows Baltic | cp1257_general_ci | 1 |
| utf32 | UTF-32 Unicode | utf32_general_ci | 4 |
| binary | Binary pseudo charset | binary | 1 |
| geostd8 | GEOSTD8 Georgian | geostd8_general_ci | 1 |
| cp932 | SJIS for Windows Japanese | cp932_japanese_ci | 2 |
| eucjpms | UJIS for Windows Japanese | eucjpms_japanese_ci | 3 |
| gb18030 | China National Standard GB18030 | gb18030_chinese_ci | 4 |
+-----+-----+-----+-----+
41 rows in set (0.02 sec)
```

查看当前连接的字符集：

```
mysql> show variables like "%char%";
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | latin1 |
| character_set_connection | latin1 |
| character_set_database | utf8 |
| character_set_filesystem | binary |
| character_set_results | latin1 |
| character_set_server | utf8 |
| character_set_system | utf8 |
| character_sets_dir | /data/tdsql_run/8812/percona-5.7.17/share/charsets/ |
+-----+-----+
```

设置当前连接相关的字符集：

```
mysql> set names utf8;
```

```
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> show variables like "%char%";
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | utf8 |
| character_set_connection | utf8 |
| character_set_database | utf8 |
| character_set_filesystem | binary |
| character_set_results | utf8 |
| character_set_server | utf8 |
| character_set_system | utf8 |
| character_sets_dir | /data/tdsql_run/8811/percona-5.7.17/share/charsets/ |
+-----+-----+
```

note：DCDB不支持设置全局参数，需要调用OSS接口设置

支持通过设置time_zone变量修改时区相关的属性

```
mysql> show variables like '%time_zone%';
```

```
+-----+-----+
```

```
| Variable_name | Value |
```

```
+-----+-----+
```

```
| system_time_zone | CST |
```

```
| time_zone | SYSTEM |
```

```
+-----+-----+
```

```
2 rows in set (0.00 sec)
```

```
mysql> create table test.tt (ts timestamp, dt datetime,c int) shardkey=c;
```

```
Query OK, 0 rows affected (0.49 sec)
```

```
mysql> insert into test.tt (ts,dt,c)values ('2017-10-01 12:12:12', '2017-10-01 12:12:12',1);
```

```
Query OK, 1 row affected (0.09 sec)
```

```
mysql> select * from test.tt;
```

```
+-----+-----+-----+
```

```
| ts | dt | c |
```

```
+-----+-----+-----+
```

```
| 2017-10-01 12:12:12 | 2017-10-01 12:12:12 | 1 |
```

```
+-----+-----+-----+
```

```
1 row in set (0.04 sec)
```

```
mysql> set time_zone = '+12:00';
```

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

```
mysql> show variables like '%time_zone%';
```

```
+-----+-----+
```

```
| Variable_name | Value |
```

```
+-----+-----+
```

```
| system_time_zone | CST |
```

```
| time_zone | +12:00 |
```

```
+-----+-----+
```

```
2 rows in set (0.00 sec)
```

```
mysql> select * from test.tt;
```

```
+-----+-----+-----+
```

```
| ts | dt | c |
```

```
+-----+-----+-----+
```

```
| 2017-10-01 16:12:12 | 2017-10-01 12:12:12 | 1 |
```

```
+-----+-----+-----+
```

```
1 row in set (0.06 sec)
```

数据类型

DCDB支持MySQL的所有数据类型，包括数字，字符，日期，空间类型，Json

数字

整型支持INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT

```
| Type | 字节数 | 最小值(有符号/无符号) | 最大值(有符号/无符号) |
```

```
| ----- | ---- | ----- | ----- |
```

```
| 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 |
```

浮点类型支持FLOAT和DOUBLE，格式

FLOAT(M,D)

或者

REAL(M,D)

或者

DOUBLE PRECISION(M,D)

定点类型支持DECIMAL和NUMERIC，格式

DECIMAL(M,D)

字符

支持如下字符类型

CHAR 和 VARCHAR Types

BINARY 和 VARBINARY Types

BLOB 和 TEXT Types

TINYBLOB，TINYTEXT，MEDIUMBLOB,MEDIUMTEXT，LONGBLOB，LONGTEXT

ENUM Type

SET Type

日期

支持如下时间类型

DATE, DATETIME, TIMESTAMP Types

TIME Type

YEAR Type

空间

支持如下空间类型

GEOMETRY

POINT

LINESTRING

POLYGON

MULTIPOINT

MULTILINESTRING

MULTIPOLYGON

GEOMETRYCOLLECTION

Json

支持存储Json格式的数据，使得对Json处理更加有效，同时又能提早检查错误

```
mysql> CREATE TABLE t1 (jdoc JSON,a int) shardkey=a;
```

```
Query OK, 0 rows affected (0.30 sec)
```

```
mysql> INSERT INTO t1 (jdoc,a)VALUES('{"key1": "value1", "key2": "value2"}',1);
```

```
Query OK, 1 row affected (0.07 sec)
```

```
mysql> INSERT INTO t1 (jdoc,a)VALUES('[1, 2,',5);
```

```
ERROR 3140 (22032): Invalid JSON text: "Invalid value." at position 6 in value for column 't1.jdoc'.
```

```
mysql> select * from t1;
```

```
+-----+-----+
```

```
| jdoc | a |
```

```
+-----+-----+
```

```
| {"key1": "value1", "key2": "value2"} | 1 |
```

```
+-----+-----+
```

```
1 row in set (0.03 sec)
```

针对json类型的orderby不支持混合类型排序，如不能将string类型和int类型做比较，同类型排序只支持数值类型，string类型，其它类型排序不处理

SQL语句语法

建表

普通的分表创建时必须最后面指定shardkey的值，该值为表中的一个字段名字，会用于后续sql的路由选择：

```
mysql> create table test1 ( a int, b int, c char(20),primary key (a,b),unique key u_1(a,c) ) shardkey=a;  
Query OK, 0 rows affected (0.07 sec)
```

由于在DCDB下，shardkey对应后端数据库的分区字段，因此必须是主键以及所有唯一索引的一部分，否则没法创建表：

```
mysql> create table test1 ( a int, b int, c char(20),primary key (a,b),unique key u_1(a,c),unique key  
u_2(b,c) ) shardkey=a;;
```

此时有一个唯一索引u_2不包含shardkey，没法创建表，会报如下错误：

```
`ERROR 1105 (HY000): A UNIQUE INDEX must include all columns in the table's partitioning function`  
因为主键索引或者unique key索引意味着需要全局唯一，而要实现全局唯一索引则必须包含shardkey字段
```

除了上面的限制外，shardkey字段还有如下要求：

1.shardkey字段的类型必须是int,bigint,smallint/char/varchar

2.shardkey字段的值不应该有中文，网关不会转换字符集，所以不同字符集可能会路由到不同的分区

3.不要update shardkey字段的值

4.shardkey=a 放在sql的最后面

5.

访问数据尽量都能带上shard

key字段，这个不是强制要求，但是不带shardkey的sql会路由到所有节点，消耗较多资源

支持建小表（广播表），此时该表在所有set中都是全量数据，这个主要方便于跨set的join操作，同时通过分布式事务保证修改操作的原子性，使得所有set的数据是完全一致的

```
mysql> create table global_table ( a int, b int key) shardkey=noshardkey_allset;  
Query OK, 0 rows affected (0.06 sec)
```

支持建立普通的表，语法和mysql完全一样，此时该表的数据全量存在第一个set中，所有该类型的表都放在第一个set中：

```
mysql> create table noshard_table ( a int, b int key);  
Query OK, 0 rows affected (0.02 sec)
```

普通sql

select：最好带上shardkey字段，否则就需要全表扫描，然后网关进行结果集聚合，影响执行效率：

```
mysql> select * from test1 where a=2;  
+-----+-----+-----+  
| a | b | c |  
+-----+-----+-----+  
| 2 | 3 | record2 |
```



```
| 2 | 4 | record3 |  
+-----+-----+-----+  
2 rows in set (0.00 sec)
```

insert/replace：字段必须包含shardkey，否则会拒绝执行该sql，因为Proxy不知道把该sql发往哪个后端数据库：

```
mysql> insert into test1 (b,c) values(4,"record3");  
ERROR 810 (HY000): Proxy ERROR:sql is too complex,need to send to only noshard table.  
Shard table insert must has field spec
```

```
mysql> insert into test1 (a,c) values(4,"record3");  
Query OK, 1 row affected (0.01 sec)
```

delete/update：为了安全考虑，执行该类sql的时候必须带有where条件，否则拒绝执行该sql命令：

```
mysql> delete from test1;  
ERROR 810 (HY000): Proxy ERROR:sql is too complex,need to send to only noshard table.  
Shard table delete/update must have a where clause
```

```
mysql> delete from test1 where a=1;  
Query OK, 1 row affected (0.01 sec)
```

聚合

DCDB支持如下全局聚合函数和全局的排序分组：sum，min，max，count，avg，order by和group by

使用方式和单机mysql一样，对于order by和group by对应的字段需要在select列表中显示指定

```
mysql> select count(*),b from test group by b;
```

```
mysql> select a,b from test order by b;
```

透传sql

在DCDB，proxy会对sql进行语法解析，会有比较严格的限制，如果用户想在某个set中执行sql，可以使用透传sql的功能：

```
mysql> select * from test1 where a in (select a from test1);  
ERROR 808 (HY000): Proxy ERROR:sql should has one shardkey  
mysql> /*set_1*/select * from test1 where a in (select a from test1);  
Empty set (0.00 sec)
```

具体语法：

```
sets:set_1,set_2 ( set名字可以通过/*proxy*/show status查询 )  
sets:allsets  
shardkey:10,支持透传sql到对应的一个或者多个set，透传到shardkey对应的set，
```

note：透传sql时，proxy不会解析sql，所以如果是往两个set进行透传写操作的话，不会使用分布式事务，极端情况下会发生不一致的问题，因此对于写操作建议一次透传一个set

JOIN

支持单个set内的join操作，单个set意味着在一个事务内的所有sql必须操作同一个set，因此必须指定shardkey字段

```
mysql> create table test1 ( a int key, b int, c char(20) ) shardkey=a;  
Query OK, 0 rows affected (1.56 sec)
```

```
mysql> create table test2 ( a int key, d int, e char(20) ) shardkey=a;  
Query OK, 0 rows affected (1.46 sec)
```

```
mysql> insert into test1 (a,b,c) values(1,2,"record1"),(2,3,"record2");
```

```
Query OK, 2 rows affected (0.02 sec)
```

```
mysql> insert into test2 (a,d,e) values(1,3,"test2_record1"),(2,3,"test2_record2");
```

```
Query OK, 2 rows affected (0.02 sec)
```

```
mysql> select * from test1 left join test2 on test1.a=test2.a;
```

```
ERROR 7015 (HY000): Proxy ERROR:sql should send to only one backend
```

```
mysql> select * from test1 left join test2 on test1.a=test2.a where test1.a=1;
```

```
+---+-----+-----+---+-----+-----+
| a | b | c | a | d | e |
+---+-----+-----+---+-----+-----+
```

```
| 1 | 2 | record1 | 1 | 3 | test2_record1 |
+---+-----+-----+---+-----+-----+
```

```
+---+-----+-----+---+-----+-----+
```

```
| 1 | 2 | record1 | 1 | 3 | test2_record1 |
+---+-----+-----+---+-----+-----+
```

```
+---+-----+-----+---+-----+-----+
```

```
1 row in set (0.00 sec)
```

支持带条件的跨set inner join

```
mysql> select * from test1 join test2 on test1.a=test2.a;
```

```
+---+-----+-----+---+-----+-----+
| a | b | c | a | d | e |
+---+-----+-----+---+-----+-----+
```

```
| 1 | 2 | record1 | 1 | 3 | test2_record1 |
+---+-----+-----+---+-----+-----+
```

```
+---+-----+-----+---+-----+-----+
```

```
| 1 | 2 | record1 | 1 | 3 | test2_record1 |
| 2 | 3 | record2 | 2 | 3 | test2_record2 |
+---+-----+-----+---+-----+-----+
```

```
| 2 | 3 | record2 | 2 | 3 | test2_record2 |
+---+-----+-----+---+-----+-----+
```

```
+---+-----+-----+---+-----+-----+
```

```
2 rows in set (0.03 sec)
```

但有如下前置条件：

- 必须是inner join
- inner join的条件必须是所有表的shardkey字段相等，支持on，using以及where格式

对于小表和noshard表相关的join，对shardkey没有限制：

```
mysql> create table noshard_table ( a int, b int key);
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> create table noshard_table_2 ( a int, b int key);
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select * from noshard_table,noshard_table_2;
Empty set (0.00 sec)
```

```
mysql> insert into noshard_table (a,b) values(1,2),(3,4);
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

```
mysql> insert into noshard_table_2 (a,b) values(10,20),(30,40);
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

```
mysql> select * from noshard_table,noshard_table_2;
+-----+---+-----+-----+
| a | b | a | b |
+-----+---+-----+-----+
| 1 | 2 | 10 | 20 |
| 3 | 4 | 10 | 20 |
| 1 | 2 | 30 | 40 |
| 3 | 4 | 30 | 40 |
+-----+---+-----+-----+
4 rows in set (0.00 sec)
```

目前不支持noshard表和shard进行join操作

分布式事务

支持分布式事务，并且对客户端透明，业务像使用单机事务一样使用。

```
begin; //开启事务
delete
update //操作数据，可以跨set
select
insert
commit; //提交事务
```

新增sql用于查询特定事务的信息：

1) select

gtid(), 获取当前分布式事务的gtid(事务的全局唯一性标识)，如果该事务不是分布式事务则返回空；

gtid的格式：

‘网关id’ - ‘网关随机值’ - ‘序列号’ - ‘时间戳’ - ‘分区号’，例如 c46535fe-b6-dd-595db6b8-25

2) select gtid_state("gtid"), 获取 "gtid" 的状态，可能的结果有：

- a) "COMMIT"，标识该事务已经或者最终会被提交
- b) "ABORT"，标识该事务最终会被回滚
- c) 空，由于事务的状态会在一个小时之后清除，因此有以下两种可能：
 - 1) 一个小时之后查询，标识事务状态已经清除
 - 2) 一个小时以内查询，标识事务最终会被回滚

注意：当commit执行超时或者失败的时候，应该等待几秒之后再调用该接口来查询事务的状态

3) 运维命令：

xa recover：向后端set发送xa recover命令，并进行汇总

xa lockwait：显示当前分布式事务的等待关系（可以使用dot命令将输出转化为等待关系图）

xa show：当前网关上正在运行的分布式事务

全局唯一字段

支持一定意义上的自增字段，保证某个字段全局唯一，但是不保证单调递增，具体使用方法如下：

创建：

```
mysql> create table auto_inc (a int,b int,c int auto_increment,d int,key auto(c),primary key p(a,d))
shardkey=d;
Query OK, 0 rows affected (0.12 sec)
```

插入：

```
mysql> insert into shard.auto_inc ( a,b,d,c) values(1,2,3,0),(1,2,4,0);
Query OK, 2 rows affected (0.05 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

```
mysql> select * from shard.auto_inc;
```

```
+---+-----+---+---+
```

```
| a | b | c | d |
```

```
+---+-----+---+---+
```

```
| 1 | 2 | 2 | 4 |
```

```
| 1 | 2 | 1 | 3 |
```

```
+---+-----+---+---+
```

```
2 rows in set (0.03 sec)
```

值得说明的是，如果在proxy调度切换、重启等过程中，自增长字段中间会有空洞，例如：

```
mysql> insert into shard.auto_inc ( a,b,d,c) values(11,12,13,0),(21,22,23,0);
Query OK, 2 rows affected (0.03 sec)
mysql> select * from shard.auto_inc;
```

```
+-----+-----+-----+-----+
```

```
| a | b | c | d |
```

```
+-----+-----+-----+-----+
```

```
| 21 | 22 | 2002 | 23 |
```

```
| 1 | 2 | 2 | 4 |
| 1 | 2 | 1 | 3 |
| 11 | 12 | 2001 | 13 |
+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

更改当前值

```
alter table auto_inc auto_increment=100
```

如果用户不指定自增值，可以通过select last_insert_id()获取

```
mysql> insert into auto_inc ( a,b,d,c) values(1,2,3,0),(1,2,4,0);
Query OK, 2 rows affected (0.73 sec)
```

```
mysql> select * from auto_inc;
+---+-----+-----+---+
| a | b | c | d |
+---+-----+-----+---+
| 1 | 2 | 4001 | 3 |
| 1 | 2 | 4002 | 4 |
+---+-----+-----+---+
2 rows in set (0.00 sec)
```

```
mysql> select last_insert_id();
+-----+
| last_insert_id() |
+-----+
| 4001 |
+-----+
1 row in set (0.00 sec)
```

note：目前

```
select last_insert_id()
```

只能跟shard表的自增字段一起使用，不支持noshard表和广播小表

数据库管理语句

状态查询

通过sql可以查看proxy的配置以及状态信息，目前支持如下命令：

```
mysql> /*proxy*/help;
```

```
+-----+-----+
| command | description |
+-----+-----+
| show config | show config from conf |
| show status | show proxy status,like route,shardkey and so on |
| set sys_log_level=N | change the sys debug level N should be 0,1,2,3 |
| set inter_log_level=N | change the interface debug level N should be 0,1 |
| set inter_time_open=N | change the interface time debug level N should be 0,1 |
| set sql_log_level=N | change the sql debug level N should be 0,1 |
| set slow_log_level=N | change the slow debug level N should be 0,1 |
| set slow_log_ms=N | change the slow ms |
| set log_clean_time=N | change the log clean days |
| set log_clean_size=N | change the log clean size in GB |
+-----+-----+
10 rows in set (0.00 sec)
```

```
mysql> /*proxy*/show config;
```

```
+-----+-----+
| config_name | value |
+-----+-----+
| version | V2R120D001 |
```



```
| mode | group shard |
| rootdir | /shard_922 |
| sys_log_level | 0 |
| inter_log_level | 0 |
| inter_time_open | 0 |
| sql_log_level | 0 |
| slow_log_level | 0 |
| slow_log_ms | 1000 |
| log_clean_time | 1 |
| log_clean_size | 1 |
| rw_split | 1 |
| ip_pass_through | 0 |
+-----+-----+
14 rows in set (0.00 sec)
```

```
mysql> /*proxy*/show status;
```

```
+-----+-----+
| status_name | value |
+-----+-----+
| cluster | group_1499858910_79548 |
| set_1499859173_1:ip |
10.49.118.165:5025;10.175.98.109:5025@1@IDC_4@0,10.231.23.241:5025@1@IDC_2@0 |
| set_1499859173_1:hash_range | 0---31 |
| set_1499911640_3:ip |
10.49.118.165:5026;10.175.98.109:5026@1@IDC_4@0,10.231.23.241:5026@1@IDC_2@0 |
| set_1499911640_3:hash_range | 32---63 |
| set | set_1499859173_1,set_1499911640_3 |
```

同时proxy增强了explain的返回结果，显示proxy修改后的sql

```
mysql> explain select * from test1;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra | info |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
| 1 | SIMPLE | test1 | ALL | NULL | NULL | NULL | NULL | 16 | | set_2,explain select * from shard.test1 |
| 1 | SIMPLE | test1 | ALL | NULL | NULL | NULL | NULL | 16 | | set_1,explain select * from shard.test1 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
2 rows in set (0.03 sec)
```

数据导入导出

支持mysqldump导出数据：

导出前设置

net_write_timeout

参数：

```
set global net_write_timeout=28800
```

```
mysqldump --compact --single-transaction --no-create-info -c
shard sbtest1 -utest -h10.231.136.34 -P3336 -ptest123
```

具体参数根据实际情况选择，如果导出的数据要导入到另外一套DCDB环境的话，必须加上-c选项

如果要导入数据的话，提供专门的导入工具，完成load data outfile对应数据的导入：

```
[tdengine@TENCENT64 ~/]$.load_data
```

```
format:./load_data mode0/mode1 proxy_host proxy_port user password db_table shardkey_index  
file field_terminate filed_enclosed
```

```
example:./load_data mode1 10.231.136.34 3336 test test123 shard.table 1 '/tmp/datafile' ' ' "
```

```
format:./load_data mode2 routers_file shardkey_index file field_terminate filed_enclosed
```

```
example:./load_data mode2 '/data/3336.xml' 1 '/tmp/datafile' ' ' "
```

note:lines should terminated by \n

note:field_terminate may have more than one char,filed_enclosed must have only one char,all can not have ',do not support escape char

预处理

Sql类型的支持：

- PREPARE Syntax
- EXECUTE Syntax

二进制协议的支持：

- COM_STMT_PREPARE
- COM_STMT_EXECUTE

例子：

```
mysql> select * from test1;
```

```
+---+-----+
```

```
| a | b |
```

```
+---+-----+
```

```
| 5 | 6 |
```

```
| 3 | 4 |
```

```
| 1 | 2 |
```

```
+---+-----+
```

```
3 rows in set (0.03 sec)
```

```
mysql> prepare ff from "select * from test1 where a=?";
```

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

```
Statement prepared
```

```
mysql> set @aa=3;
```

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

```
mysql> execute ff using @aa;
```

```
+---+-----+
```

```
| a | b |
```

```
+---+-----+
```

```
| 3 | 4 |
```

```
+---+-----+
```

```
1 row in set (0.06 sec)
```

子查询

DCDB对于子查询这块支持比较有限，目前只支持带shardkey的derived table

```
mysql> select a from (select * from test1) as t;
```

```
ERROR 7012 (HY000): Proxy ERROR:sql should has one shardkey
```

```
mysql> select a from (select * from test1 where a=1) as t;  
+---+  
| a |  
+---+  
| 1 |  
+---+  
1 row in set (0.00 sec)
```

两级分区

DCDB只用HASH方式进行数据拆分，不利于删除特定条件的数据，如流水类型，删除旧的数据，为了解决这个问题，可以使用两级分区。

DCDB支持range和list格式的两级分区，具体建表语法和mysql分区语法类似

range支持类型

DATE , DATETIME , TIMESTAMP

支持year , month , day函数，函数为空和day函数一样

TINYINT, SMALLINT, MEDIUMINT, INT (INTEGER), and BIGINT

支持year , month , day函数，此时传入的值转换为年月日，然后和分表信息对比

函数为空则直接使用该int值和分表信息对比

例子：

如果hired是date类型，则查询插入对应的值格式为 '20160101 10:20:20' ,20160101

```
CREATE TABLE employees_int (  
id INT key NOT NULL,  
fname VARCHAR(30),
```

```
lname VARCHAR(30),
hired date,
separated DATE NOT NULL DEFAULT '9999-12-31',
job_code INT,
store_id INT
)
shardkey=id
PARTITION BY RANGE ( month(hired) ) (
PARTITION p0 VALUES LESS THAN (199102),
PARTITION p1 VALUES LESS THAN (199603),
PARTITION p2 VALUES LESS THAN (200101)
);
```

如果hired是int类型，则查询插入对应的值格式为1474961034，proxy首先会转换成对应的date格式，20160927，然后和分表信息对比

```
CREATE TABLE employees_int (
id INT key NOT NULL,
fname VARCHAR(30),
lname VARCHAR(30),
hired int,
separated DATE NOT NULL DEFAULT '9999-12-31',
job_code INT,
store_id INT
)
shardkey=id
PARTITION BY RANGE ( month(hired) ) (
PARTITION p0 VALUES LESS THAN (199102),
PARTITION p1 VALUES LESS THAN (199603),
PARTITION p2 VALUES LESS THAN (200101)
);
```

list支持类型

DATE , DATETIME , TIMESTAMP , 支持年月日函数

TINYINT, SMALLINT, MEDIUMINT, INT (INTEGER), and BIGINT

CHAR, VARCHAR, BINARY, and VARBINARY.

```
CREATE TABLE customers_1 (  
  first_name VARCHAR(25),  
  last_name VARCHAR(25),  
  street_1 VARCHAR(30),  
  street_2 VARCHAR(30),  
  city VARCHAR(15),  
  renewal DATE  
) shardkey=first_name  
PARTITION BY LIST (city) (  
  PARTITION pRegion_1 VALUES IN('1', '2', '3'),  
  PARTITION pRegion_2 VALUES IN('4', '5', '6'),  
  PARTITION pRegion_3 VALUES IN('7', '8', '9'),  
  PARTITION pRegion_4 VALUES IN('10', '11', '12')  
);
```

注意：分区使用的是小于符号，因此如果要存储当年的数据的话（2017），需要创建<2018的分区，用户只需创建到当前的时间分区，DCDB会自动增加后续分区，默认往后创建3个分区，以YEAR为例，DCDB会自动创建2018，2019，2020的分区，后续也会自动增减

函数支持

Control Flow Functions

Name	Description
-----	-----

CASE	Case operator
IF()	If/else construct
IFNULL()	Null if/else construct
NULLIF()	Return NULL if expr1 = expr2

String 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
LOAD_FILE()	Load the named file
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 |

| MATCH | Perform full-text search |

| MID() | Return a substring starting from the specified position |

| NOT LIKE | Negation of simple pattern matching |

| NOT REGEXP | Negation of REGEXP |

| 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 |

| SOUNDEX() | Return a soundex string |

| SOUNDS LIKE | Compare sounds |

| 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 |

| WEIGHT_STRING() | Return the weight string for a string |

Numeric Functions and Operators

| Name | Description |

| ----- | ----- |

| ABS() | Return the absolute value |

| ACOS() | Return the arc cosine |

| ASIN() | Return the arc sine |

| ATAN() | Return the arc tangent |

| ATAN2(), ATAN() | Return the arc tangent of the two arguments |

| CEIL() | Return the smallest integer value not less than the argument |

| CEILING() | Return the smallest integer value not less than the argument |

| CONV() | Convert numbers between different number bases |

| COS() | Return the cosine |

| COT() | Return the cotangent |

| CRC32() | Compute a cyclic redundancy check value |

| DEGREES() | Convert radians to degrees |

| DIV | Integer division |

| / | Division operator |

| EXP() | Raise to the power of |

| FLOOR() | Return the largest integer value not greater than the argument |

| LN() | Return the natural logarithm of the argument |

| LOG() | Return the natural logarithm of the first argument |

| LOG10() | Return the base-10 logarithm of the argument |

| LOG2() | Return the base-2 logarithm of the argument |

| - | Minus operator |

| MOD() | Return the remainder |

| %, MOD | Modulo operator |

| PI() | Return the value of pi |

| + | Addition operator |

| POW() | Return the argument raised to the specified power |

| POWER() | Return the argument raised to the specified power |

RADIANS()	Return argument converted to radians
RAND()	Return a random floating-point value
ROUND()	Round the argument
SIGN()	Return the sign of the argument
SIN()	Return the sine of the argument
SQRT()	Return the square root of the argument
TAN()	Return the tangent of the argument
*	Multiplication operator
TRUNCATE()	Truncate to specified number of decimal places
-	Change the sign of the argument

Date and 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 |

| 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
YEARWEEK()	Return the year and week

Aggregate (GROUP BY) Functions

Name	Description
AVG()	Return the average value of the argument
COUNT()	Return a count of the number of rows returned
MAX()	Return the maximum value
MIN()	Return the minimum value
SUM()	Return the sum

Bit Functions and Operators

Name	Description
BIT_COUNT()	Return the number of bits that are set
&	Bitwise AND
~	Bitwise inversion
|	Bitwise OR
^	Bitwise XOR
<<	Left shift
>>	Right shift

Cast Functions and Operators

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

读写分离

DCDB支持三种模式的读写分离。

- DCDB开启语法解析的配置，通过语法解析过滤出用户的select读请求，默认把读请求直接发给备机；（该功能风险较大，需提交工单方能开启）
- 通过增加slave注释标记，将指定的sql发往备机。即在sql中添加/slave/这样的标记，该sql会发送给备机；
- 由只读帐号发送的请求会根据配置的属性发给备机

错误码和错误信息

proxy增加如下错误编码

```
enum proxy_error
{
    ER_PROXY_GRAM_ERROR_BEGIN=800,

    ER_PROXY_SANITY_ERROR, /*sql类型错误*/
    ER_PROXY_BAD_COMMAND, /*sql命令不支持*/
    ER_PROXY_SQL_NOT_SUPPORT, /*DCDB不支持类型*/
    ER_PROXY_SQLUSE_NOT_SUPPORT, /*DCDB不支持该用法*/
    ER_PROXY_ERROR_SHARDKEY, /*建表时一级二级分区键选择有问题，如两者要不一样，是表中某一列*/

    ER_PROXY_ERROR_SUB_SHARDKEY, /*暂时不用*/
}
```

```
ER_PROXY_PRE_DEAL_ERROR, /*join 不符合规则*/
ER_PROXY_SHADKEY_ERROR, /*复杂sql , 如derived table需要包含shardkey*/
ER_PROXY_COMBINE_SQL_ERROR, /*重组sql有问题 , 如没有对应的二级分区表*/
ER_PROXY_SQL_ERROR, /*一般的sql错误*/
ER_PROXY_ONE_SET, /*count ( distinct ) 必须发到一个set上*/
ER_PROXY_STRICT_ERROR, /*暂时不用*/
ER_PROXY_TRANSACTION_ERROR, /*事务相关的错误*/
```

```
ER_PROXY_GRAM_ERROR_END,
ER_PROXY_SYSTEM_ERROR_BEGIN=900,
```

```
ER_PROXY_SLICING,
ER_PROXY_NO_DEFAULT_SET, /*set为空*/
ER_PROXY SOCK_ADDRESS_ERROR, /*set地址为空*/
ER_PROXY SOCK_ERROR, /*连接后端数据库失败*/
ER_PROXY_STATUS_ERROR, /*group状态出错*/
ER_PROXY_UNKNOWN_ERROR, /*unknown错误*/
```

```
ER_PROXY_ERROR_END
```

```
};
```

其中900以上为系统错误，会通过监控平台进行告警

从单机实例导入到分布式实例

1.数据导入概述

由于分布式数据库的分布式架构对用户透明，一般情况下，只需要预先建好表结构。可以使用mysqldump、或其他Navicat、SQLyog等MySQL客户端进行迁移。笔者对迁移步骤总结如下：

1. 第一步：准备导入导出环境
2. 第二步：导出源表的表结构和数据
3. 第三步：修改建表语句并在目的表中创建表结构
4. 第四步：导入数据

2.准备导入导出环境

准备迁移数据前，您需要准备好入下环境：

- 云服务器
 - 建议配置CPU 2核，内存8GB，磁盘500GB以上（取决于数据量大小）
 - linux
 - 安装MySQL客户端
 - 如果您迁移的数据量较小(<10GB)，也可以通过外网（互联网）直接导入，无需准备
- 分布式云数据库DCDB：
 - 根据预期选择大小，并根据源库字符集，表名大小写，innodb_page_size大小进行初始化
 - 创建帐号，该帐号建议开启全局所有权限
 - 必要时开启外网IP

3.导出源表的表结构和数据

3.1 演示环境

- 操作库：caccts
- 操作表：t_acct_water_0
- 源库：单实例mysql
- 目标库：DCDB for Percona、MariaDB

3.2 在源库导出表结构

通过命令

```
mysqldump -u username -p password -d dbname tablename > tablename.sql
```

导出表结构

//命令实例

```
mysqldump -utest -ptest1234 -d -S /data/4003/prod/mysql.scok caccts t_acct_water_0 > table.sql
```

3.2 在源库导出表数据

通过命令

```
mysqldump -c -u username -p password dbname tablename > tablename.sql
```

导出表结构

//命令实例

```
mysqldump -c -t -utest -ptest1234 -S /data/4003/prod/mysql.scok caccts t_acct_water_0 > data.sql
```

注意：导出数据必须通过mysqldump工具导出，并且加上 -c 参数，因为这样导出的数据行都带有列名字段，不带列名字段的sql会被DCDB for Percona、MariaDB拒绝掉。-t 参数的意义是不导出表结构，只导出表数据。

3.3 上传文件至云服务器某目录

上传前，您开启CVM外网访问地址，并请

参考CVM上传文件说明：[Linux机器通过SCP上传文件](#)，即可将文件上传,您至少需要上传刚刚导出的：

- 表结构sql：table.sql
- 数据sql：data.sql

4.修改建表语句并在目的表中创建表结构

打开刚刚导出的表结构文件 table.sql，参考如下格式语句添加主键和shardkey，并另存为 tablenew.sql。请注意[shardkey选择技巧和注意事项](#)：

```
CREATE TABLE ( 列名称1 数据类型, 列名称2 数据类型, 列名称3 数据类型, .... , PRIMARY KEY('列名称n') )  
ENGINE=INNODB DEFAULT CHARSET=xxxx shardkey=keyname
```

注意：必须要设置主键，必须指定shardkey，必须注意表名大小问题，建议删除多余注释，否则建表可能不成功。

```
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
```

5.导入数据

5.1 连接DCDB for Percona、MariaDB实例

在CVM上使用

```
mysql -u username -p password -h IP -P port
```

登录MySQL服务器；然后使用

```
use dbname
```

进入数据库

注意：您可能需要先创建库。

5.2 导入表结构

使用刚刚上传的文件，用

```
source
```

命令导入数据

1. 先导入表结构：

```
source /文件路径/tablenew.sql；
```

2. 再导入数据：

```
source /文件路径/data.sql；
```

3. 校验导入情况：

```
select count(*) from tablename;
```

注意：需先导入建表语句，再导入数据。也可以通过mysql的source命令直接导入sql。

6.其他方案

整体来说，只要能够在导入数据前，在目的表先创建对应的表结构（需指定shardkey），就可以比较顺利的导入数据。

如何选择DCDB实例配置和分片配置

DCDB 选型概述

DCDB 由分片（sharding）组成，分片的规格和分片数量决定了 DCDB 实例的处理能力。理论上讲：

- DCDB 实例读写并发性能 = $\sum$ （某规格分片性能 * 某规格分片数量）
- DCDB 实例事务性能 = $\sum$ （某规格分片事务性能 70% 某规格分片数量）

因此，分片规格越高、分片数量越多，实例的处理能力越强。而分片性能，主要与 CPU / 内存 相关，并以 QPS 为基础衡量指标，我们在分片性能说明章节，给出了大致性能指标。

DCDB 分片规格的选择

DCDB 分片规格的选择，主要从三个方面需求来决定：1、性能需求；2、容量需求；3、其他要求。

性能需求：通过预判至少6个月的性能规模和可能增长，您可以确定您分布式实例所需总 CPU / 内存 规模

容量需求：通过预判至少1年的容量规模和可能增长，您可以确定您分布式实例所需总 磁盘 规模

其他要求：我们建议一个分片至少存储5000W行数据，并考虑到业务中所需的[广播表、单表](#)，和节点内join等业务需求。

注意：我们建议您先确保让单个分片配置较大，而分片数量较少。

综合上述来看，我们预估您们可能有如下几种业务特点，我们推荐策略如下：

- 使用 DCDB
做功能性测试，且对性能没有特别要求：2个分片，每个分片配置为：内存/磁盘：2GB/25GB。
- 业务发展初期，总数据
规模较小但增长快的选型：2个分片，每个分片配置为：内存/磁盘：16GB/200GB。
- 业务发展稳定，根据业务实际情况选型：4个分片，每个分片配置等于：当前业务峰值*增长率/4

DCDB 分片性能测试

数据库基准性能测试为sysbench 0.5 工具修改说明：对sysbench自带的otlp脚本做了修改，读写比例修改为1：1，并通过执行测试命令参数oltp_point_selects和oltp_index_updates来控制读写比例，本文测试用例的均采用4个select点，1个update点，读写比例保持4：1。

vCPU (核)	内存(GB)	存储空间(GB)	数据集(GB)	客户端数	单客户端并发数	QPS	TPS
1	2GB	100GB	46GB	1	128	1880	351
2	4GB	200GB	76GB	1	128	3983	797
2	8GB	200GB	142GB	1	128	6151	1210
4	16GB	400GB	238GB	1	128	10098	2119
4	32GB	700GB	238GB	2	128	20125	3549
8	64GB	1T	378GB	2	128	37956	7002
12	96GB	1.5T	378GB	3	128	51026	10591
16	120GB	2T	378GB	3	128	81050	15013
24	240GB	3T	567GB	4	128	96891	17698
48	480GB	6T	567GB	6	128	140256	26599

此处TPS为单机TPS，并非测试的是分布式事务的TPS；

根据运营策略要求，当前DCDB的（部分）实例都采用闲时超用技术，所以您可能在您的监控中看到CPU利用率超过100%的情况。

从分布式实例导入到分布式实例

由于分布式数据库到分布式数据库的数据导入方案与一般情况不同，笔者这里使用mysqldump对分布式实例导入数据到分布式实例的步骤总结如下：

第一步：安装MariaDB版本的mysqldump

购买云linux版本的云服务器，使用

```
yum install mariadb-server
```

即可安装。

第二步：导出源库的表结构

```
mysqldump --no-tablespaces --default-character-set=utf8 --hex-blob --skip-add-drop-table -c  
--skip-lock-tables --skip-opt --skip-triggers --skip-comments --skip-tz-utc --skip-extended-insert  
--skip-disable-keys --skip-quote-names --no-data -uusername -ppassword -hxxx.xxx.xxx.xxx  
-Pxxxxxx  
dbname | grep -v '^\\^*\\!40' > schema.sql;
```

备注：--default-character-set=utf8 应根据您源表实际情况设定。

-uusername 是有权限的帐号（-u是关键字）

-ppassword 是密码（-p是关键字）

-hxxx.xxx.xxx.xxx -Pxxxxxx 是数据库实例的IP和端口。

dbname 代表db的名字

使用grep过滤掉/!40.../的语句，是因为这些语句在DCDB里面不支持。

第二步：导出源库的表数据


```
mysqldump --no-tablespaces --default-character-set=utf8 --hex-blob --skip-add-drop-table -c  
--skip-lock-tables --skip-opt --skip-triggers --skip-comments --skip-tz-utc --skip-extended-insert  
--skip-disable-keys --skip-quote-names --no-create-info -uandelwu -pandelwu -hxxx.xxx.xxx.xxx  
-Pxxxxx  
dbname | grep -v '^\\.*!40' > data.sql;
```

备注：--default-character-set=utf8 应根据您源表实际情况设定。

-uusername 是有权限的帐号（-u是关键字）

-ppassword 是密码（-p是关键字）

-hxxx.xxx.xxx.xxx -Pxxxxx 是数据库实例的IP和端口。

dbname 代表db的名字

使用grep过滤掉!/40.../的语句，是因为这些语句在DCDB里面不支持。

第三步：在目标库创建db

```
mysql --default-character-set=utf8 -uusername -ppassword -hxxx.xxx.xxx.xxx -Pxxxxx -e "create  
database dbname;"
```

备注：--default-character-set=utf8 应根据您目标表实际情况设定。

-uusername 是有权限的帐号（-u是关键字）

-ppassword 是密码（-p是关键字）

-hxxx.xxx.xxx.xxx -Pxxxxx 是数据库实例的IP和端口。

dbname 代表db的名字

第四步：在目标库上导入表结构+

```
mysql --default-character-set=utf8 -uusername -ppassword -hxxx.xxx.xxx.xxx -Pxxxxx dbname <
schema.sql
```

备注：--default-character-set=utf8 应根据您目标表实际情况设定。

-uusername 是有权限的帐号（-u是关键字）

-ppassword 是密码（-p是关键字）

-hxxx.xxx.xxx.xxx -Pxxxxx 是数据库实例的IP和端口。

dbname 代表db的名字

第五步：在目标库上导入表数据

```
mysql --default-character-set=utf8 -uusername -ppassword -hxxx.xxx.xxx.xxx -Pxxxxx dbname <
data.sql
```

备注：如果源表中使用了自增字段，并且导入的时候出现 “Column 'xx' specified twice” 的错误，则需要对schema.sql做处理去掉自增字段的反引号(`cat schema.sql | tr "`" " " > schema_tr.sql`)，然后drop database，使用处理后的schema_tr.sql重复步骤3-5的操作。