TDSQL for MySQL

Development Guide

Product Documentation

Contents

Development Guide
InnoDB
Overview
Use Limits
Compatibility
Connecting Database
Join and Subquery
Sequence
Frequently Used DML Statements
Read/Write Separation
Distributed Transaction
Creating Tables
Connection Protection
Globally Unique Field
Exporting/Importing Data
Database Management Statements
Passthrough SQL
Preprocessing
Subpartitioning
Error Codes and Messages

Development Guide

InnoDB

Overview

Last updated：2021-01-08 17:49:54

TDSQL for MySQL is highly compatible with MySQL protocols and syntax. However, due to architectural differences, TDSQL has certain restrictions on SQL. In order to better take advantage of the distributed architecture, we recommend that you follow the suggestions below.

The TDSQL instance is capable of horizontal scaling, making it suitable for scenarios with massive amounts of data. Its features are described as follows:

- Supports flexible read/write separation
- Supports global operations including ORDER BY, GROUP BY, and LIMIT
- Supports aggregate functions including SUM, COUNT, AVG, MIN, and MAX
- Supports cross-node (set) join operations and subqueries
- Supports pre-processing protocols
- Supports the global unique field and sequence
- Supports distributed transactions
- Supports two-level partitioning
- Provides specific SQL statements for querying configuration and status of the entire cluster

The TDSQL instance supports three different types of tables:

- **Sharded table**: this is a horizontally sharded table. It is a complete logical table from the business perspective, but the backend distributes the data to different node sets according to the hash value of the shardkey.
- **Non-sharded table**: this is also known as a noshard table, which does not need to be specifically sharded or processed. Currently, the TDSQL instance stores this type of table in the first physical node set by default.
- **Broadcast table**: this is based on small table broadcasting technology. After a table is set as a broadcast table, all operations of the table will be broadcast to all node sets, where each set has all the data of the table. This is often used as a configuration table of a business system.

Note：

- In a TDSQL instance, if two tables have the same **shardkey**, the rows corresponding to the same shardkey in the two tables are definitely stored in the same physical node set. This type of scenario is often called groupshard, which greatly improves the processing efficiency of statements such as join queries.

- Since non-sharded tables are placed on the first set by default, if a large number of non-sharded tables are created in a TDSQL instance, the load of the first set will be too high.
- In most cases, we recommend that you create sharded tables in a TDSQL instance.

Use Limits

Last updated：2022-05-25 15:44:18

Feature Limits

- Custom functions, events, and tablespaces are not supported
- Views, stored procedures, triggers, and cursors are not supported
- Foreign keys and self-built partitions are not supported
- Compound statements such as BEGIN END,and LOOP are not supported
- SQL syntax related to master/slave sync is not supported

Limits on small syntax

DDL

- CREATE TABLE ... SELECT is not supported
- CREATE TEMPORARY TABLE is not supported
- CREATE/DROP/ALTER SERVER/LOGFILE GROUP are not supported
- ALTER is not supported for renaming shardkey, but it can be used to change the type
- RENAME is not supported

DML

- SELECT INTO OUTFILE/INTO DUMPFILE/INTO var_name are not supported
- query_expression_options is not supported, such as HIGH_PRIORITY/STRAIGHT_JOIN/SQL_SMALL_RESULT/SQL_BIG_RESULT/SQL_BUFFER_RESULT/SQL_CACHE/SQL_NO_CACHE/SQL_CALC_FOUND_ROWS
- INSERT/REPLACE without column name are not supported
- ORDER BY/LIMIT cannot be used for global DELETE/UPDATE (Version greater than or equal to 1.14.4 can be supported)
- UPDATE/DELETE without WHERE condition are not supported
- LOAD DATA/XML are not supported
- DELAYED/LOW_PRIORITY cannot be used in SQL statements
- INSERT ... SELECT is not supported (Version greater than or equal to 1.14.4 can be supported)
- References and operations on variables in SQL are not supported, such as SET @c=1, @d=@c+1; SELECT @c, @d
- index_hint is not supported

- HANDLER/DO are not supported

SQL management

- ANALYZE/CHECK/CHECKSUM/OPTIMIZE/REPAIR TABLE are not supported, which should be performed with passthrough syntax
- CACHE INDEX are not supported
- FLUSH is not supported
- KILL is not supported
- LOAD INDEX INTO CACHE is not supported
- RESET is not supported
- SHUTDOWN is not supported
- SHOW BINARY LOGS/BINLOG EVENTS is not supported
- The combination of SHOW WARNINGS/ERRORS and LIMIT/COUNT is not supported

Other limits

Up to 5,000 tables can be created by default. If you need to create more tables, please submit a ticket.

Compatibility

Last updated：2021-05-06 17:51:16

Language Structures

TDSQL for MySQL supports all literal values used by MySQL, including:

```
String Literals
Numeric Literals
Date and Time Literals
Hexadecimal Literals
Bit-Value Literals
Boolean Literals
NULL Values
```

String literals

A string literal is a sequence of bytes or characters, enclosed within either single quote `'` or double quote `"` characters. Currently, TDSQL does not support the ANSI_QUOTES SQL mode, so things enclosed with double quote `"` characters are always interpreted as string literals instead of identifiers.

TDSQL does not support character set introducers, i.e., the format of `[_charset_name]'string' [COLLATE collation_name]` .

TDSQL supports the following escape characters:

```
\0: an ASCII NUL (X'00') character
\': a single quote (') character
\": a double quote (") character
\b: a backspace character
\n: a newline (linefeed) character
\r: a carriage return character
\t: a tab character
\z: ASCII 26 (Ctrl+Z)
\\: a backslash (\) character
\%: \%
\_: _
```

Numeric literals

Numeric literals include integer, decimal, and floating-point literals.

Integers are represented as a sequence of digits which may include `.` as a decimal separator. A numeric literal may be preceded by `-` or `+` to indicate a negative or positive value, respectively.

Exact-value numeric literals can be represented as follows: `1, .2, 3.4, -5, -6.78, +9.10`.

Scientific notation examples: `1.2E3, 1.2E-3, -1.2E3, -1.2E-3`.

Date and time literals

TDSQL supports the following DATE formats:

```
'YYYY-MM-DD' or 'YY-MM-DD'
'YYYYMMDD' or 'YYMMDD'
YYYYMMDD or YYMMDD

For example, '2012-12-31', '2012/12/31', '2012^12^31', '2012@12@31', '20070523',
and '070523'.
```

TDSQL supports the following DATETIME and TIMESTAMP formats:

```
'YYYY-MM-DD HH:MM:SS' or 'YY-MM-DD HH:MM:SS'
'YYYYMMDDHHMMSS' or 'YYMMDDHHMMSS'
YYYYMMDDHHMMSS or YYMMDDHHMMSS

For example, '2012-12-31 11:30:45', '2012^12^31 11+30+45', '2012/12/31 11*30*45',
'2012@12@31 11^30^45', and 19830905132800.
```

Hexadecimal literals

TDSQL supports the following formats:

```
X'01AF'
X'01af'
x'01AF'
x'01af'
0x01AF
0x01af
```

Bit-value literals

TDSQL supports the following formats:

```
b'01'
B'01'
0b01
```

Boolean literals

The constants TRUE and FALSE evaluate to 1 and 0, respectively. The constant names can be written in any lettercase.

```
mysql> SELECT TRUE, true, FALSE, false;
+------+------+-------+-------+
| TRUE | TRUE | FALSE | FALSE |
+------+------+-------+-------+
| 1 | 1 | 0 | 0 |
+------+------+-------+-------+
1 row in set (0.03 sec)
```

NULL values

The NULL value means no data. NULL can be written in any lettercase. A synonym is \N (case-sensitive).
Be aware that the NULL value is different from values such as 0 for numeric types or the empty string (`''`) for string types.

Character Sets and Time Zones

TDSQL supports all character sets and collations supported by MySQL.

```
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)
```

View character sets of the current connection:

```
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/ |
+--------------------------+--------------------------------------------------------
+
```

Set character sets of the current connection:

```
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：

TDSQL does not support setting global parameters which can only be set by calling frontend APIs.

Support modifying time zone attributes by setting the `time_zone` variable.

```
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 1
2:12:12',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)
```

Data Types

TDSQL supports all data types supported by MySQL, including numbers, strings, date and time, spatial types, and JSON.

Numbers

For integer types, INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, and BIGINT are supported.

Type	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

For floating-point types, FLOAT and DOUBLE are supported in the format of FLOAT(M,D), REAL(M,D), or DOUBLE PRECISION(M,D).

For fixed-point types, DECIMAL and NUMERIC are supported in the format of DECIMAL(M,D).

Strings

TDSQL supports the following string types:

```
CHAR and VARCHAR
BINARY and VARBINARY
BLOB and TEXT
TINYBLOB, TINYTEXT, MEDIUMBLOB, MEDIUMTEXT, LONGBLOB, and LONGTEXT
ENUM
SET
```

Date and time

TDSQL supports the following date and time types:

```
DATE, DATETIME, and TIMESTAMP
TIME
YEAR
```

Spatial data

TDSQL supports the following spatial types:

```
GEOMETRY
POINT
LINESTRING
POLYGON

MULTIPOINT
MULTILINESTRING
MULTIPOLYGON
GEOMETRYCOLLECTION
```

JSON

TDSQL supports storing data in JSON format, making JSON processing more efficient while ensuring that errors can be detected in advance.

Note：

You cannot sort JSON values of different types. For example, you cannot compare JSON values of STRING type with those of INTEGER type. For JSON values of the same type, only numbers and strings can be compared and sorted. In the table created by the following SQL statements, the statement `select * from t1 order by t1->"$.key2"` is not supported because the column sorted by the ORDER BY clause contains both numbers and strings.

```
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 (0.00 sec)

mysql> INSERT INTO t1 (jdoc,a)VALUES('{"key1": "value1", "key2": 2}',2);

mysql> INSERT INTO t1 (jdoc,a)VALUES('[1, 2,',5);
ERROR 3140 (22032): Invalid JSON text: "Invalid value." at position 6 in value fo
r column 't1.jdoc'.

mysql> select * from t1;
+---------------------------------------+---+
| jdoc | a |
+---------------------------------------+---+
| {"key1": "value1", "key2": "value2"} | 1 |
| {"key1": "value1", "key2": 2} | 2 |
+---------------------------------------+---+
2 rows in set (0.00 sec)
```

Supported Functions

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

Connecting Database

Last updated：2021-03-02 17:16:43

Connecting with Client

TDSQL for MySQL supports a connection method compatible with MySQL. It can be connected using the IP address, port number, username, and password, as shown below:

```
mysql -hxxx.xxx.xxx.xxx -Pxxxx -uxxx -pxxx -c
```

Note：

TDSQL for MySQL does not support clients earlier than version 4.0 and compression protocols. We recommend that you add the `-c` option when using the client, so that you can use advanced features.

Connecting with PHP MySQLi

To connect to TDSQL for MySQL in PHP, you need to enable the MySQLi extension. A demo is as follows:

```
header("Content-Type:text/html;charset=utf-8");
$host="10.10.10.10"; //Instance proxy_host_ip
$user="test"; //Instance user
$pwd="test"; //Instance password
$db="aaa"; //Database name
$port="15002"; //proxy_host port number
$sqltool=new MySQLli($host,$user,$pwd,$db,$port);
//Other necessary code
$sqltool->close();
echo "ok"."\n";
```

Connecting with JDBC

You can run the following code to connect to TDSQL for MySQL with JDBC:

```
private final String USERNAME = "test";
private final String PASSWORD = "123456";
private final String DRIVER = "com.mysql.jdbc.Driver";
private final String URL = "jdbc:mysql://10.10.10.10:3306?userunicode=true&charac
terEncoding=utf8mb4";
private Connection connection;
private PreparedStatement pstmt;
private ResultSet resultSet;
```

Other Connection Methods

TDSQL for MySQL supports other connection methods compatible with MySQL, such as Navicat and ODBC.

Join and Subquery

Last updated：2021-01-11 15:30:36

In TDSQL for MySQL, data is split horizontally across nodes. To improve database performance, we recommend that you prioritize the optimization of table structures and SQL statements, and try not to manipulate data across nodes.

Recommended SQL Statements

For multiple sharded tables with a WHERE condition of the same shardkey

```
MySQL > select * from test1 join test2 where test1.a=test2.a;
+---+------+---------+---+------+---------------+
| a | b | c | a | d | e |
+---+------+---------+---+------+---------------+
| 1 | 2 | record1 | 1 | 3 | test2_record1 |
| 2 | 3 | record2 | 2 | 3 | test2_record2 |
+---+------+---------+---+------+---------------+
2 rows in set (0.00 sec)

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 | 2 | 3 | test2_record2 |
+---+------+---------+------+------+---------------+
1 row in set (0.00 sec)

MySQL> select * from test1 where test1.a in (select a from test2);
+---+------+---------+
| a | b | c |
+---+------+---------+
| 1 | 2 | record1 |
| 2 | 3 | record2 |
+---+------+---------+
2 rows in set (0.00 sec)

MySQL> select a, count(1) from test1 where exists (select * from test2 where test
2.a=test1.a) group by a;
+---+----------+
| a | count(1) |
+---+----------+
| 1 | 1 |
| 2 | 1 |
+---+-----------+
2 rows in set (0.00 sec)

MySQL> select distinct count(1) from test1 where exists (select * from test2 wher
e test2.a=test1.a) group by a;
+-----------+
| count(1) |
+-----------+
| 1 |
+-----------+
1 row in set (0.00 sec)

MySQL> select count(distinct a) from test1 where exists (select * from test2 wher
e test2.a=test1.a);
+-------------------+
| count(distinct a) |
+-------------------+
| 2 |
+-------------------+
1 row in set (0.00 sec)
```

For non-sharded tables

```
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)
```

For broadcast tables

```
MySQL> create table global_test(a int key, b int)shardkey=noshardkey_allset;
Query OK, 0 rows affected (0.00 sec)

MySQL> insert into global_test(a, b) values(1,1),(2,2);
Query OK, 2 rows affected (0.00 sec)

MySQL> select * from test1, global_test;
+---+------+---------+---+------+
| a | b | c | a | b |
+---+------+---------+---+------+
| 1 | 2 | record1 | 1 | 1 |
| 2 | 3 | record2 | 1 | 1 |
| 1 | 2 | record1 | 2 | 2 |
| 2 | 3 | record2 | 2 | 2 |
+---+------+---------+---+------+
4 rows in set (0.00 sec)
```

For derived tables where the subquery has a shardkey

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

Complex SQL Statements

For SQL statements that cannot be optimized into the recommended statements, cross-node data interaction is required, resulting in a relatively low performance.
Including:

- Queries with subqueries

- Join queries for multiple tables which have different shardkeys or are of different types (for example, non-sharded tables and sharded tables)

In such complex queries, the query conditions will be sent to shards to extract data that actually participates in the query from backend databases. The extracted data will be stored in a local temporary table and then be computed.

Therefore, you need to explicitly specify the query conditions of the tables to avoid performance degradation due to the extraction of large amounts of data.

```
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_record
2");
Query OK, 2 rows affected (0.02 sec)

mysql> select * from test1 join test2 on test1.b=test2.d;
+---+------+---------+---+------+---------------+
| a | b | c | a | d | e |
+---+------+---------+---+------+---------------+
| 2 | 3 | record2 | 1 | 3 | test2_record1 |
| 2 | 3 | record2 | 2 | 3 | test2_record2 |
+---+------+---------+---+------+---------------+
2 rows in set (0.00 sec)

MySQL> select * from test1 where exists (select * from test2 where test2.a=test1.
b);
+---+------+---------+
| a | b | c |
+---+------+---------+
| 1 | 2 | record1 |
+---+------+---------+
1 row in set (0.00 sec)
```

TDSQL also supports many complex UPDATE, DELETE, and INSERT operations.

Note that these operations are based on SELECT operations. As data needs to be extracted into a temporary table on the gateway, we recommend that you explicitly specify the query conditions to avoid performance degradation due to the extraction of large amounts of data.

In addition, the gateway will not lock the extracted data by default, which is slightly different from the official MySQL. If you need to lock the data, you can modify the proxy configuration.

```
MySQL [th]> update test1 set test1.c="record" where exists(select 1 from test2 wh
ere test1.b=test2.d);
Query OK, 1 row affected (0.00 sec)

MySQL [th]> update test1, test2 set test1.b=2 where test1.b=test2.d;
Query OK, 1 row affected (0.00 sec)

MySQL [th]> insert into test1 select cast(rand()*1024 as unsigned), d, e from tes
t2;
Query OK, 2 rows affected (0.00 sec)

MySQL [th]> delete from test1 where b in (select b from test2);
Query OK, 6 rows affected (0.00 sec)

MySQL [th]> delete from test2.* using test1 right join test2 on test1.a=test2.a w
here test1.a is null;
Query OK, 2 rows affected (0.00 sec)
```

Sequence

Last updated：2022-06-02 15:18:31

The sequence keyword syntax is compatible with MariaDB/Oracle, but a sequence must be globally incremental and unique. The specific usage is as follows:

Note：

- When using a sequence in TDSQL for MySQL, the keyword must be prefixed with `tdsql_`, and the proxy version must be later than 1.19.5-M-V2.0R745D005. You can view the proxy version by running the `/*Proxy*/show status` statement. If the proxy is on an earlier version, you can submit a ticket for upgrade.
- Currently, sequence is used to guarantee the global uniqueness of the distributed transaction, which has a relatively low performance and is therefore only suitable for scenarios with low concurrency.

Creating a sequence requires the `CREATE SEQUENCE` system permission. The creation syntax is as follows:

```
CREATE TDSQL_SEQUENCE sequence name
[START WITH n]
[{TDSQL_MINALUE/ TDSQL_MAXMINVALUE n| TDSQL_NOMAXVALUE}]
[TDSQL_INCREMENT BY n]
[{TDSQL_CYCLE|TDSQL_NOCYCLE}]
```

Creating Sequence

```
create tdsql_sequence test.s1 start with 12 tdsql_minvalue 10 maxvalue 50000 tdsq
l_increment by 5 tdsql_nocycle
create tdsql_sequence test.s2 start with 12 tdsql_minvalue 10 maxvalue 50000 tdsq
l_increment by 1 tdsql_cycle
```

- The above SQL statements include six parameters: start value, minimum value, maximum value, increment, buffer size, and whether to cycle, all of which should be positive integers.
- The default values of these parameters are as follows: start value (1), minimum value (1), maximum value (LONGLONG_MAX-1), increment (1), and whether to cycle (0).

Deleting Sequence

```
drop tdsql_sequence test.s1
```

Querying Sequence

```
show create tdsql_sequence test.s2
```

Using Sequence

Getting next value

```
select tdsql_nextval(test.s2)
select next value for test.s2
```

```
mysql> select tdsql_nextval(test.s1);
+-----+
| 12 |
+-----+
| 12 |
+-----+
1 row in set (0.18 sec)
mysql> select tdsql_nextval(test.s2);
+-----+
| 12 |
+-----+
| 12 |
+-----+
1 row in set (0.13 sec)
mysql> select tdsql_nextval(test.s1);
+-----+
| 17 |
+-----+
| 17 |
+-----+
1 row in set (0.01 sec)
mysql> select tdsql_nextval(test.s2);
+-----+
| 13 |
+----+
| 13 |
+----+
1 row in set (0.00 sec)
mysql> select next value for test.s1;
+----+
| 22 |
+----+
| 22 |
+----+
1 row in set (0.01 sec)
```

Using nextval in INSERT and other statements

```
mysql> select * from test.t1;
+----+------+
| a | b |
+----+------+
| 11 | 2 |
+----+------+
1 row in set (0.00 sec)
mysql> insert into test.t1(a,b) values(tdsql_nextval(test.s2),3);
Query OK, 1 row affected (0.01 sec)
mysql> select * from test.t1;
+----+------+
| a | b |
+----+------+
| 11 | 2 |
| 14 | 3 |
+----+------+
2 rows in set (0.00 sec)
```

The last value is needed to join relevant data. If it has never been obtained with the `nextval` command, 0 will be returned.

```
select tdsql_lastval(test.s1)
select tdsql_previous value for test.s1;
```

```
mysql> select tdsql_lastval(test.s1);
+----+
| 22 |
+----+
| 22 |
+----+
1 row in set (0.00 sec)
```

```
mysql> select tdsql_previous value for test.s1;
+----+
| 22 |
+----+
| 22 |
+----+
1 row in set (0.00 sec)
```

Set the next value for the sequence, which must be greater than the current value; otherwise, 0 will be returned.

```
select tdsql_setval(test.s2,1000,bool use) // `use` is 1 by default, indicating t
hat the value of 1,000 has been used and will not be included next time. If this
is 0, the next return will start from 1,000.
```

If the next value of the sequence is smaller than the current value, the system will make no response.

```
mysql> select tdsql_nextval(test.s2);
+----+
| 15 |
+----+
| 15 |
+----+
1 row in set (0.01 sec)
mysql> select tdsql_setval(test.s2,10);
+---+
| 0 |
+---+
| 0 |
+---+
1 row in set (0.03 sec)
mysql> select tdsql_nextval(test.s2);
+----+
| 16 |
+----+
| 16 |
+----+
```

If the set next value is larger than the current value, the set value will be successfully returned.

```
mysql> select tdsql_setval(test.s2,20);
+----+
| 20 |
+----+
| 20 |
+----+
1 row in set (0.02 sec)
```

```
mysql> select tdsql_nextval(test.s2);
+----+
| 21 |
+----+
| 21 |
+----+
1 row in set (0.01 sec)
```

Forcibly set the next sequence value (you can set a value smaller than the current value).

```
select tdsql_resetval(test.s2,1000)
```

If the forced setting is successful, the set value will be returned, from which the next sequence value will start.

```
mysql> select tdsql_resetval(test.s2,14);
+----+
| 14 |
+----+
| 14 |
+----+
1 row in set (0.00 sec)
mysql> select tdsql_nextval(test.s2);
+----+
| 14 |
+----+
| 14 |
+----+
1 row in set (0.01 sec)
```

Note that some sequence keywords are prefixed with `TDSQL_` :

```
TDSQL_CYCLE
TDSQL_INCREMENT
TDSQL_LASTVAL
TDSQL_MINVALUE
TDSQL_NEXTVAL
TDSQL_NOCACHE
TDSQL_NOCYCLE
TDSQL_NOMAXVALUE
TDSQL_NOMINVALUE
TDSQL_PREVIOUS
TDSQL_RESTART
TDSQL_REUSE
TDSQL_SEQUENCE
TDSQL_SETVAL
```

Frequently Used DML Statements

Last updated：2020-11-09 17:14:18

select

We recommend including the shardkey field whose hash value allows the direct routing of SQL request through the proxy to the database instance. Otherwise, the proxy needs to send requests to all database instances in the cluster and aggregates the results, which degrades performance:

```
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

Include the shardkey field to avoid errors, because the proxy is unable to determine the target backend database for SQL.

```
mysql> insert into test1 (b,c) values(4,"record3");
ERROR 810 (HY000): Proxy ERROR:sql is too complex,need to send to only noshard ta
ble.
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

For security reasons, include the `where` condition to avoid errors.

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

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

Read/Write Separation

Last updated：2021-09-22 10:49:41

TDSQL for MySQL instances support read-write separation in the following modes:

- A comment flag such as `/*slave*/` can be added to send specified SQL statements to the secondary server.

> Note：
> Comment flags including `/*slave:slaveonly*/`, `/*slave:20*/`, and `/*slave:slaveonly,20*/` are also supported, where the value indicates the delay requirement that the secondary server should satisfy, and `slaveonly` indicates that the query will not be sent to the primary server if there is no eligible secondary server.

- Requests sent by read-only accounts are sent to the secondary server according to the configured attributes.

Distributed Transaction

Last updated：2021-01-11 15:29:22

In TDSQL for MySQL, a transaction usually involves data of multiple physical nodes. Such transactions are called distributed transactions.
TDSQL supports XA and non-XA distributed transaction protocols. By default, TDSQL (kernel version 5.7 or later) supports distributed transactions which are imperceptible to the client and as easy-to-use as non-distributed transactions.
Distributed transactions in TDSQL adopt a two-phase commit protocol (2PC) to ensure the atomicity and consistency of transactions, and support such isolation levels as `Read committed` , `Repeatable read` , and `Serializable` .

Non-XA Distributed Transactions

```
begin; # Start a transaction
... # Cross-set INSERT, DELETE, UPDATE, SELECT, and other non-DDL operations
commit; # Commit the transaction
```

XA Distributed Transactions

XA distributed transactions refer to cross-instance transactions.

```
xa begin ''; # Start an XA transaction. The transaction identifier is generated b
y the system, so you can pass in an empty string.
... # Cross-set INSERT, DELETE, UPDATE, SELECT, and other non-DDL operations
select gtid(); # Get the XA transaction identifier, which is assumed as 'xid' in
the following statements
xa prepare 'xid'; # Prepare the transaction
xa commit/rollback 'xid'; # Commit or roll back the transaction
```

New Transactions APIs

- `select gtid()` : this API is used to get the globally unique identifier of the distributed transaction. If no value is returned, the transaction is not a distributed transaction.

  - The format of non-XA distributed transaction identifier is: 'Gateway ID' - 'Proxy random value' - 'Serial number' - 'Timestamp' - 'Partition number', such as c46535fe-b6-dd-595db6b8-25.
  - The format of XA distributed transaction identifier is: 'ex' - 'Gateway ID' - 'Proxy random value' - 'Serial number' - 'Timestamp' - 'Partition number', such as ex-c46535fe-b6-dd-595db6b8-25.

- `select gtid_state("Globally unique identifier of the distributed transaction")` : this API is used to get the transaction status (in 3 seconds by default) after an exception occurs in committing the transaction. The following status may be returned:
  - COMMIT: indicates that the transaction has been or is going to be committed.
  - ABORT: indicates that the transaction is going to be rolled back.
  - Empty value: the transaction status will be cleared after an hour, so there are two possibilities:
    - For a query after an hour, it indicates that the transaction status has been cleared.
    - For a query within an hour, it indicates that the transaction is going to be rolled back.

- `xa boost 'Globally unique identifier of the distributed transaction'` : after an exception occurs in committing a non-XA transaction, the transaction will be automatically committed or rolled back by the backend component in a certain period of time (30 seconds by default). If you are unwilling to wait for such a long time, you can repeatedly call this API to make the system commit or roll back the transaction in a timely manner. This API will return the status of the transaction (i.e., committed or rolled-back).

- `xa lockwait` : this API is used to show the wait-for relationship between the distributed transactions. You can convert it to a graph using the dot tool.

- `xa show` : this API is used to show transactions that are active on the proxy.

Creating Tables

Last updated：2021-05-06 14:28:11

Creating a Sharded Table

When creating a sharded table, you must specify the value of shardkey at the end of the SQL statement. The value of shardkey is a field name in the table and will be used for subsequent SQL routing.

```
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)
```

In a TDSQL instance, the shardkey corresponds to the partition field of the backend database, so it must be part of all primary keys and unique indexes; otherwise, the table cannot be created.

Use case : an error occurs when there are multiple unique indexes.

```
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;;
```

According to the above SQL statement, there is a unique index `u_2` that does not include the shardkey field, so the table cannot be created and the following error message will be displayed:

```
ERROR 1105 (HY000): A UNIQUE INDEX must include all columns in the table's partit
ioning function
```

The index on the primary key or unique key has to be globally unique. To do so, the index must include the shardkey field.

In addition to the above restrictions, the shardkey field has the following requirements:

- The type of the shardkey field must be int, bigint, smallint, char, or varchar.
- The value of the shardkey field should not contain Asian characters, as the proxy will not convert the character set, and different character sets may be routed to different partitions.
- Do not update the value of the shardkey field.
- `shardkey=a` should be placed at the end of the SQL statement.
- We recommend that you specify the value of shardkey in SQL statements when accessing data. This is not mandatory, but SQL statements without shardkey will be routed to all nodes and thus consume more resources.

Creating a Broadcast Table

You can create small tables (broadcast tables). Each set has all the data of a small table, which makes it easier to perform cross-set joins. In addition, distributed transactions ensure the atomicity of the modification operations, so that the data in all sets is exactly identical.

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

Creating a Non-sharded Table

You can create non-sharded tables using the same syntax as MySQL. All the data in this type of tables is stored in the first set.

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

Connection Protection

Last updated：2020-10-26 10:32:44

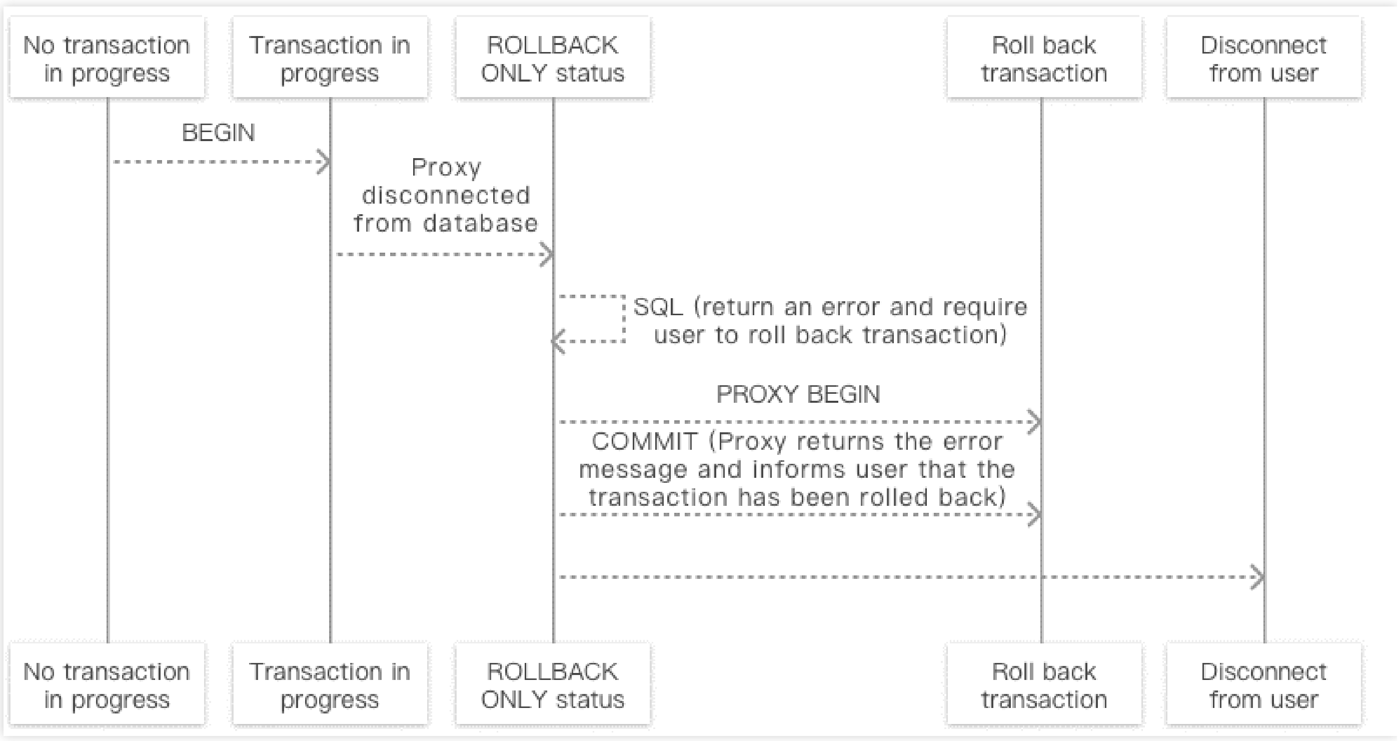
Connection protection keeps the connection intact between the client and the proxy when the proxy is disconnected from the backend database, so in cases such as when the backend database has an exception, the proxy will not be affected.

If the proxy is disconnected from the database while executing SQL, the proxy will close all connections related to the SQL (except the connection between itself and the client) and display an error:

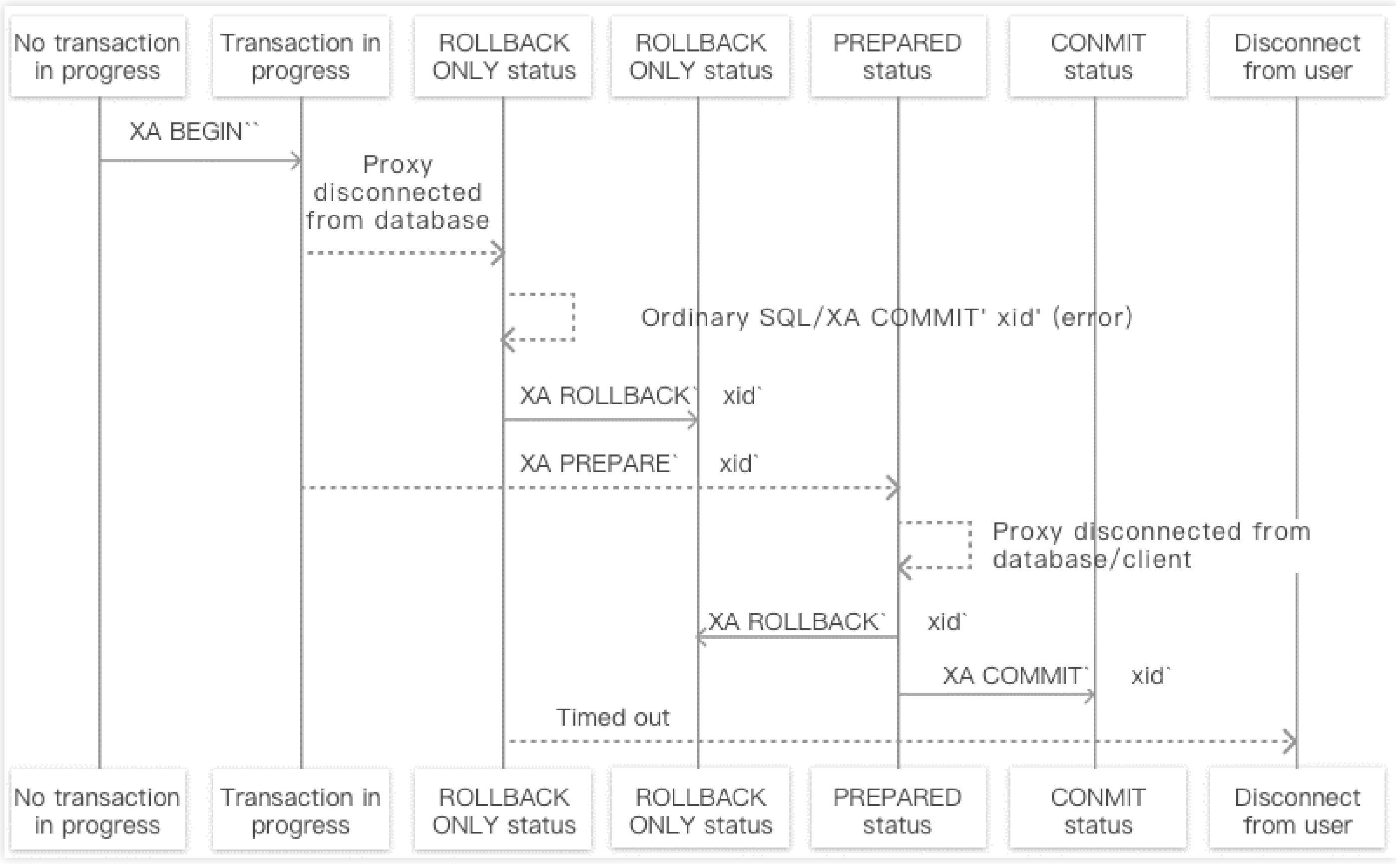
```
ER_PROXY_TRANSACTION_ERROR // Transaction in progress
ER_PROXY_CONN_BROKEN_ERROR // No transaction in progress
```

Solutions

- If the proxy is disconnected from the backend database and your session is in a "transaction in progress" status, this can be fixed as follows (all the error codes below are ER_PROXY_TRANSACTION_ERROR):


No transaction in progress
Transaction in progress
ROLLBACK ONLY status
Roll back transaction
Disconnect from user
BEGIN
Proxy disconnected from database
SQL (return an error and require user to roll back transaction)
PROXY BEGIN
COMMIT (Proxy returns the error message and informs user that the transaction has been rolled back)

- If the proxy is disconnected from the backend database and your session is in an "XA transaction in progress" status, this can be fixed as follows (all the error codes below are ER_PROXY_TRANSACTION_ERROR):



Timeout Configuration

Note：

If the proxy disconnects from the backend database during the execution of a transaction and the transaction fails to be rolled back before the timeout period elapses, the proxy will disconnect from the client.

The timeout parameter is configured in the proxy configuration file as follows:

```
<server_close timeout="60"/>
```

Globally Unique Field

Last updated：2020-11-09 17:20:18

The `auto_increment` keyword can be used to create a global unique auto-incremental field which is definitely global unique but may not be monotonically increasing. You can use `auto_increment` as instructed below.

Create

```
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)
```

Insert

```
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)
```

If a process such as switching or restarting occurs, there will be holes in the auto-increment fields, for example:

```
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)
```

Change the current value:

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

Use `select last_insert_id()` to query the last inserted value of the auto-incremental field:

```
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)
```

Currently, you can use `select last_insert_id()` to query the auto-incremental field in a sharded table or broadcast table. Such an operation is unsupported in a non-sharded table.

Exporting/Importing Data

Last updated：2022-05-25 15:37:01

Exporting Data

TDSQL for MySQL supports exporting data with mysqldump. Before export, you need to set the `net_write_timeout` parameter (`set global net_write_timeout=28800`) in the TDSQL for MySQL Console. Note that command line utilities are not authorized by TDSQL for MySQL, so you can set the parameter only in the console.

```
mysqldump --compact --single-transaction --no-create-info -c db_name table_name -
utest -h10.xx.xx.34 -P3336 -ptest123
```

Note：

- The `db` and `table` parameters should be specified based on the actual situation. If the exported data is to be imported into another set of TDSQL for MySQL environment, the `-c` option must be added.
- Export account needs to have `select on *.*` permissions.

Importing Data

TDSQL for MySQL provides a professional tool to import data specified by `load data outfile` . This tool shards a source file into multiple files according to the shardkey routing rules, and passes each file through to the corresponding backend database.

Download the tool.

```
[tdengine@TENCENT64 ~/]$./load_data
format:./load_data mode0/mode1 proxy_host proxy_port user password db_table shard
key_index file field_terminate filed_enclosed
example:./load_data mode1 10.xx.xx.10 3336 test test123 shard.table 1 '/tmp/dataf
ile' ' ' ''
```

Note：

- The source file must use '\n' as a line break.
- `mode0` means that the source file is sharded but will not be imported, which is usually used for debugging. To import data, use `mode1` .
- `shardkey_index` starts from 0. If the shardkey is the second field, `shardkey_index` should be 1.

Database Management Statements

Last updated：2021-03-12 11:22:10

Query status

You can view the configuration and status of the proxy through SQL. The following commands are currently supported:

- `/*proxy*/help;`
- `/*proxy*/show config;`
- `/*proxy*/show status;`

Note：

If you're logging to MySQL client, add `-c` to the codes, such as `mysql -hxxx.xxx.xxx.xxx -Pxxxx -uxxx -pxxx -c`.

Sample codes:

```
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@0,10.231.2
3.241:5025@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@0,10.231.2
3.241:5026@1@IDC_2@0 |
| set_1499911640_3:hash_range | 32---63 |
| set | set_1499859173_1,set_1499911640_3 |
```

Meanwhile, the proxy enhances the returned result of explain, showing the SQL statement modified by the proxy:

```
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 sel
ect * from shard.test1 |
| 1 | SIMPLE | test1 | ALL | NULL | NULL | NULL | NULL | 16 | | set_1,explain sel
ect * from shard.test1 |
+------+-------------+--------+------+---------------+------+---------+------+----
--+-------+-----------------------------------------+
2 rows in set (0.03 sec)
```

Passthrough SQL

Last updated：2021-12-13 10:04:30

TDSQL for MySQL parses SQL syntax, which restricts SQL execution. If you want to execute SQL statements supported by MySQL but not by TDSQL on a set, you can pass through the SQL statements.

Note：

- The proxy will not parse an SQL statement that is passed through. If it is a passthrough write operation to two sets, distributed transactions will not be used, which may cause inconsistency in extreme cases. Therefore, we recommend you pass through only to one set in a single write operation.
- To ensure that the passthrough syntax takes effect, add the `-c` parameter when connecting to MySQL.

```
MySQL [test]> repair table test.t1;
ERROR 664 (HY000): Proxy ERROR:SQL is too complex, only applicable to noshard tab
le: Shard table do not support repair
MySQL [test]> /*sets:allsets*/repair table test.t1;
+---------+--------+----------+----------+------------------+
| Table | Op | Msg_type | Msg_text | info |
+---------+--------+----------+----------+------------------+
| test.t1 | repair | status | OK | set_1544429866_3 |
| test.t1 | repair | status | OK | set_1544429718_1 |
+---------+--------+----------+----------+------------------+
2 rows in set (0.01 sec)
```

Syntax:

- sets:set_1,set_2: it is used to specify the sets to which SQL statements will be passed through. The set name can be queried by `/*proxy*/show status` .
- sets:allsets: it indicates that SQL statements will be passed through to all sets.
- shardkey:10: it indicates that SQL statements will be passed through to the set according to the specified shardkey.
- shardkey_hash:10: it indicates that SQL statements will be passed through to the set where its specified hash value is 10. If `shardkey_hash` is set to 0, SQL statements will be passed through to the first set.

Preprocessing

Last updated：2020-09-14 11:06:46

Supported SQL syntax:

- PREPARE Syntax
- EXECUTE Syntax

Supported binary protocols:

- COM_STMT_PREPARE
- COM_STMT_EXECUTE

Sample:

```
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)
```

Subpartitioning

Last updated：2022-10-11 16:32:03

TDSQL for MySQL currently supports two-level sharding in range and list formats, where the specific table creation syntax is similar to the sharding syntax in MySQL.

Two-Level Sharding Syntax

The syntax for creating a sharded table with a hash at level 1 and a list at level 2 is as follows:

```
MySQL [test]> CREATE TABLE customers_1 (
first_name VARCHAR(25) key,
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('Beijing', 'Tianjin', 'Shanghai'),
PARTITION pRegion_2 VALUES IN('Chongqing', 'Wulumuqi', 'Dalian'),
PARTITION pRegion_3 VALUES IN('Suzhou', 'Hangzhou', 'Xiamen'),
PARTITION pRegion_4 VALUES IN('Shenzhen', 'Guangzhou', 'Chengdu')
);
```

The syntax for creating a sharded table with a range at level 1 and a list at level 2 is as follows:

```
MySQL [test]> CREATE TABLE tb_sub_r_l (
id int(11) NOT NULL,
order_id bigint NOT NULL,
PRIMARY KEY (id,order_id))
PARTITION BY list(order_id)
(PARTITION p0 VALUES in (2121122),
PARTITION p1 VALUES in (38937383))
TDSQL_DISTRIBUTED BY RANGE(id) (s1 values less than (100),s2 values less than (10
00));
Query OK, 0 rows affected, 1 warning (0.35 sec)
```

Supported range types

- DATE, DATETIME, and TIMESTAMP.

- `year` , `month` , and `day` functions are supported. If the function is empty, it will be defaulted to the `day` function.
- TINYINT, SMALLINT, MEDIUMINT, INT, and BIGINT.
- `year` , `month` , and `day` functions are supported. The value entered is converted to year, month, and day and then compared against the sharded table information.

Supported list types

- DATE, DATETIME, and TIMESTAMP.
- `year` , `month` , and `day` functions are supported.
- TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT and VARCHAR.

Alarm

- Refrain from using the TIMESTAMP type as the shardkey, because it is subject to the time zone and can only specify a time value before the year of 2038.
- If the shardkey is `char` or `varchar` type, its length is better to be below 255.

Use Cases and Suggestions

Use one-level sharded tables for businesses as much as possible.

- Before use, design the table structure reasonably according to the business scenario in the long run. Two-Level sharding is suitable for scenarios where DDL changes are not required for a long time after the table structure is created, while the sharded data needs to be cleaned and trimmed regularly, such as log transaction tables.
- Design the granularity of two-level sharding reasonably. We recommend you refrain from using too fine-grained two-level sharding; otherwise, too many subtables will be created. For example, you should shard transaction tables by month instead of by day or by hour, so that there will not be too many data files in the file system.
- When performing an SQL query on a level-2 sharded table, include the key values of level-1 and level-2 sharding in the query conditions as much as possible, so as to eliminate the need to open many data files for search during query execution.
- When performing a JOIN query on a level-2 sharded table, if the query conditions don't include the key values of level-1 and level-2 sharding, the operation performance will be low, which is not recommended.
- The primary key or unique index of the table should include the shardkey; otherwise, the data uniqueness cannot be guaranteed.

Error Codes and Messages

Last updated：2020-10-26 10:32:45

The following error codes have been added for the proxy:

```
#define ER_PROXY_GRAM_ERROR_BEGIN 600

#define ER_PROXY_SANITY_ERROR 601 // "Sanity error: %s"
#define ER_PROXY_GS_NOT_SUPPORT 602 // Unsupported SQL type
#define ER_PROXY_ORDERBY_INDEX_NEG 603 // order by index is negative
#define ER_PROXY_ORDERBY_INDEX_TOO_BIG 604 // order by index is too big
#define ER_PROXY_ORDERBY_TYPE_UNSUPPORT 605 // ORDER BY is not supported
#define ER_PROXY_GROUPBY_INDEX_NEG 606 // group by index is negative
#define ER_PROXY_GROUPBY_INDEX_TOO_BIG 607 // group by index is too big
#define ER_PROXY_GROUPBY_TYPE_UNSUPPORT 608 // GROUP BY is not supported
#define ER_PROXY_GET_AUTO_ID_FAILED 609 // Failed to obtain auto-increment ID
#define ER_PROXY_TEANS_ROLLED_BACK 610 // The transaction has been rolled back
#define ER_PROXY_ONE_SET 611 // The current SQL statement should have been sent t
o one backend but it has not
#define ER_PROXY_CLIENT_HS_ERROR 612 // An error occurred while parsing the clien
t handshake packet
#define ER_PROXY_ACCESS_DENIED_ERROR 613 // The length of readu_auth_switch_resul
t is not 20, which should not happen
#define ER_PROXY_TRANS_NOT_ALLOWED 614 // Command not allowed in the transaction
#define ER_PROXY_TRANS_READ_ONLY 615 // Command not allowed in the read-only tran
saction
#define ER_PROXY_TRANS_ERROR_DIFFENT_SET 616 // In the non-XA transaction, the re
ad-only SQL uses multiple backends
#define ER_PROXY_STRICT_ERROR 617 // In the strict mode, only one set can be modi
fied at a time
#define ER_PROXY_SC_TOO_LONG 618 // The backend is disconnected for a long time a
nd the connection is closed
#define ER_PROXY_START_TRANS_FAILED 619 // Failed to start a new XA transaction
#define ER_PROXY_SC_RETRY 620 // The server is closed. Please retry the previous
SQL statement.
#define ER_PROXY_SC_TRANS_IN_ROLLBACK_ONLY 621 // The server is closed. The curre
nt transaction is being rolled back.
#define ER_PROXY_SC_COMMIT_LATER 622 // The server is closed. The transaction wil
l be committed later.
#define ER_PROXY_SC_ROLLBACL_LATER 623 // The server is closed. The transaction w
ill be rolled back later.
#define ER_PROXY_SC_IN_COMMIT_OR_ROLLBACK 624 // The server was closed when the t
ransaction was being committed/rolled back
#define ER_PROXY_SC_NEED_ROLLBACK 625 // The server is closed. The current transa
ction needs to be rolled back.
```

```
#define ER_PROXY_SC_STATE_WILL_ROLLBACK 626 // The server is closed. Rollback wil
l be performed.
#define ER_PROXY_XA_UNSUPPORT 627 // Command is not supported by XA currently.
#define ER_PROXY_XA_INVALID_COMMAND 628 // The XA command is invalid.
#define ER_PROXY_XA_GTID_INIT_ERROR 629 // GTID log initialization failed
#define ER_PROXY_XA_GET_SET_IP_PORT_FAILED 630 // Failed to obtain the set addres
s
#define ER_PROXY_XA_UPDATE_GTID_LOG_FAILED 631 // Failed to update GTID log
#define ER_PROXY_MYSQL_PARSER_ERROR 632 // Failed to parse the embedded database
SQL
#define ER_PROXY_ILLEGAL_ID 633 // Kill ID is invalid
#define ER_PROXY_NOT_SUPPORT_CURSOR 634 // CURSOR_TYPE_READ_ONLY is currently not
supported
#define ER_PROXY_UNKNOWN_PREPARE_HANDLER 635 // The executed prepare is unknown
#define ER_PROXY_SET_PARA_FAIL 636 // Set parameters failed
#define ER_PROXY_SUBPARTITION_DEAY 637 // An error occurred while handling subpar
titions
#define ER_PROXY_NO_SUBPARTITION_ROUTE 638 // No routing information was obtained
for the subpartitioned table
#define ER_PROXY_LOCK_MORE_TABLE 639 // Only one subpartitioned table can be lock
ed at a time
#define ER_PROXY_GET_ROUTER_LOCK_FAIL 640 // Failed to obtain the route lock
#define ER_PROXY_PART_NAME_EMPTY 641 // Partition name is empty
#define ER_PROXY_SUB_PART_TABLE_IS_NONE 646 // There is no subpartition
#define ER_PROXY_PART_TYPE_DENY 643 // The subpartition type is not supported
#define ER_PROXY_PART_NAME_ILLEGAL 644 // The partition name is invalid
#define ER_PROXY_DROP_ALL_PARTITION_FAIL 645 // Failed to delete all partitions.
Please try deleting the table.
#define ER_PROXY_GET_OLD_PART_NUM_FAIL 646 // Failed to obtain the number of shar
ds of the table
#define ER_PROXY_EMPTY_SQL 647 // Empty SQL, which will not be returned to the cl
ient
#define ER_PROXY_ERROR_SHARDKEY 648 // The shardkey must be a specified column
#define ER_PROXY_ERROR_SUB_SHARDKEY 649 // Shardkey for subpartitioning failed
#define ER_PROXY_SQLUSE_NOT_SUPPORT 650 // The proxy does not support this usage
#define ER_PROXY_DBFW_WHITE_LIST_DENY 651 // Not in the allowlist and blocked by
the firewall
#define ER_PROXY_DBFW_DENY 652 // Blocked by the firewall
#define ER_PROXY_INCORRECT_ARGS 653 // The "stmt" parameter is incorrect
#define ER_PROXY_SYSTABLE_UNSUPPORT_NON_READ_SQL 654 // Non-read-only SQL stateme
nts cannot access system tables
#define ER_PROXY_TABLE_NOT_EXIST 655 // The table does not exist
#define ER_PROXY_SHARD_JOIN_UNSUPPORT_TYPE 656 // Shard join is currently not sup
ported
#define ER_PROXY_RECURSIVE_JOIN_DENY 657 // Recursive join is not supported
#define ER_PROXY_JOIN_INTERNAL_ERROR 658 // The join operation is exceptional
#define ER_PROXY_SQL_TOO_COMPLEX 659 // The SQL statement is too complex and grou
```

```
pshard is currently not supported
#define ER_PROXY_INVALID_ARG_FOR_GTID_STATE 660 // The "gtid_state()" parameter i
s invalid
#define ER_PROXY_CANT_SET_GLOBAL_AUTOCOMMIT_GS 661 // Global autocommit cannot be
set in groupshard
#define ER_PROXY_INVALID_VALUE_FOR_AUTOCOMMIT 662 // The "autocommit" value is in
valid
#define ER_PROXY_XID_ERROR 663 // xid is invalid
#define ER_PROXY_XID_GENERAT_FAILED 664 // xid cannot be specified by the user
#define ER_PROXY_CANT_EXEC_IN_INTER_TRANS 665 // "The command cannot be executed
in internal transction"
#define ER_PROXY_XID_TIME_ERROR 666 // "Unexpected time part of xid"
#define ER_PROXY_XID_TIMEDIFF_TOO_LONG 667 // "timediff > 1800s, it's not safe to
execute boost"
#define ER_PROXY_SAVEPOINT_NOT_EXIST 668 // The SAVEPOINT does not exist
#define ER_PROXY_SC_TRANS_IN_ROLLED 669 // The transaction has been rolled back a
s the server is closed
#define ER_PROXY_CANT_BOOST_IN_TRANS 670 // SQLCOM_BOOST is not allowed in the tr
ansaction
#define ER_PROXY_TRANS_EXPECTED 671 // "A transaction is expected, this maybe a b
ug"
#define ER_PROXY_EXTERNAL_TRANS 672 // An external XA transaction cannot be execu
ted
#define ER_PROXY_AUTO_INC_FAIL 673 // "Deal auto inc failed"
#define ER_PROXY_CHECK_JOIN_FAIL 674 // "Check join failed"
#define ER_PROXY_TABLE_TYPE_NOT_MATCH 675 // "Do not support shard-table operatio
ns in noshard instance"
#define ER_PROXY_UNSUPPORT_NS_IN_INSERT 676 // "Do not support noshard and noshar
d_allset in insert sql"
#define ER_PROXY_ALTER_SEQ_ID_FAIL 677 // Alter seq id failed
#define ER_PROXY_ALTER_ID_ILLEGAL 678 // Alter seq id is illegal
#define ER_PROXY_CANT_CHANGE_STEP 679 // "Current table use zk to get auto inc, d
o not support to change step: \'%s\'"
#define ER_PROXY_ALTER_STEP_FAIL 680 // Alter step failed
#define ER_PROXY_TOO_MUCH_TABLES 681 // The maximum number of tables has been rea
ched
#define ER_PROXY_TABLE_EXISTED 682 // The table already exists
#define ER_PROXY_CREATE_STABLE_FAILED 683 // Failed to create the sharded table,
as complex SQL statements cannot be used to create a sharded table
#define ER_PROXY_DDL_DENY 684 //The DDL operation is not allowed
#define ER_PROXY_SHADKEY_ERROR 685 // SQL should not relate to subpartition table
s
#define ER_PROXY_NO_SK 686 // reject nosk
#define ER_PROXY_COMBINE_SQL_KEY 687 // Something went wrong
#define ER_PROXY_GET_SK_ERROR 688 // Failed to get the shardkey
#define ER_PROXY_SHOW_FAILED 689 // The "proxy show" command is incorrect
#define ER_PROXY_SET_FAILED 690 // The "proxy set" command is incorrect
```

```
#define ER_PROXY_SET_FAILED 690 // The SQL statement format is incorrect
#define ER_PROXY_UNLOCK_ROUTER_FAIL 692 // Failed to disable route lock
#define ER_PROXY_LOCK_ROUTER_FAIL 693 // Failed to enable route lock
#define ER_PROXY_PROXY_CMD_FAIL 694 // The "/*proxy*/" command is not supported
#define ER_PROXY_PROCESS_RULE_FILE_FAILED 695 // dump_error
#define ER_PROXY_GET_AUTO_NUM_ERROR 696 // Failed to obtain the auto-increment va
lue
#define ER_PROXY_SEQUENCE_NOT_EXIST 697 // The sequence does not exist
#define ER_PROXY_SEQUENCE_ERROR 698 // The sequence is invalid
#define ER_PROXY_SEQUENCE_ALREADY_EXIST 699 // The sequence already exists
#define ER_PROXY_SQL_RETRY 700 // The SQL statement has not been committed or rol
led back
#define ER_PROXY_XA2PC_ABORT 701 // 2PC failed and the transaction will be rolled
back
#define ER_PROXY_XA2PC_COMMIT 702 // 2PC failed and it will be committed later
#define ER_PROXY_XA2PC_UNCERTAIN 703 // 2PC failed with unknown result
#define ER_PROXY_KILL_ERROR 704 // Failed to kill
#define ER_PROXY_TRACE_DENY705 // The SQL statement cannot be executed in the tra
ce mode
#define ER_PROXY_SQL_IMCOMPLETE 706 // The transaction is in the incomplete statu
s
#define ER_PROXY_SHARDKEY_HASH_ERROR 709// Shardkey hash error

#define ER_PROXY_GRAM_ERROR_END 799

// system error -----------------------------------------------------------------
------
#define ER_PROXY_SYSTEM_ERROR_BEGIN 900

#define ER_PROXY_SLICING 901 // The slice is modified and possibly being scaled,
causing the current SQL statement to be rejected
#define ER_PROXY_NO_DEFAULT_SET 902 // The set is empty
#define ER_PROXY_GET_ADDRESS_FAILED 903 // Initialization has not been completed
yet. Failed to obtain the backend address. Please try again later.
#define ER_PROXY_SQL_SIZE_ERROR_IN_GET_CANDIDATE_ADDRESS 904 // An error occurred
while obtaining the backend address (the number of destination backend is incorre
ct)
#define ER_PROXY_GET_ADDRESS_ERROR 905 // An error occurred while obtaining the b
ackend address
#define ER_PROXY_CANDIDATE_ADDRESS_EMPTY 906 // No backend address has been obtai
ned
#define ER_PROXY_CANT_GET_SOCK 907 // Failed to obtain the socket
#define ER_PROXY_GET_SET_SOCK_FAIL 908 // Failed to obtain the socket
#define ER_PROXY_CONNECT_ERROR 909 // Backend connection failed
#define ER_PROXY_NO_SQL_ASSIGN_TO_SET 910 // An exception occurred while sending
SQL statements
#define ER_PROXY_STATUS_ERROR 911 // The group is in an exceptional status and di
```

```
sconnected
#define ER_PROXY_CONN_BROKEN_ERROR 912 // The server is closed and the SQL statem
ent is in an exceptional status
#define ER_PROXY_UNKNOWN_ERROR 913 // Unknown proxy error
#define ER_PROXY_ALL_SLAVES_UNAVAILABLE 914 // All secondary servers are unavaila
ble
#define ER_PROXY_ALL_SLAVES_CHANGE 915 // The secondary server has an exception

#define ER_PROXY_ERROR_END 916
```

Note：

Error code 900 and higher are system errors, which will be alarmed through the Cloud Monitor platform.