

Mysql-SQL&Function

1. [零 参考资料 3](#_Toc49681880)

[一 Functions and Operators 3](#_Toc49681881)

[1 概述 3](#_Toc49681882)

[2 运算符 3](#_Toc49681883)

[2.1 概述 3](#_Toc49681884)

[2.2 比较运算符 4](#_Toc49681885)

[3 Control Flow Functions 6](#_Toc49681886)

[4 数字相关 Functions & Operator 7](#_Toc49681887)

[4.1 计算运算符 7](#_Toc49681888)

[4.2 Function 8](#_Toc49681889)

[5 Date and Time function 10](#_Toc49681890)

[6 String Function and operators 11](#_Toc49681891)

[7 Aggregate Functions 13](#_Toc49681892)

[8 Other Functions and Operators 13](#_Toc49681893)

[二 SQL Statements 16](#_Toc49681894)

[1 Data Definition Statements（DDL） 16](#_Toc49681895)

[1.1 ALTER DATABASE 16](#_Toc49681896)

[1.2 ALTER EVENT 16](#_Toc49681897)

[1.3 ALTER FUNCTION 16](#_Toc49681898)

[1.4 ALTER INSTANCE 16](#_Toc49681899)

[1.5 ALTER LOGFILE GROUP 16](#_Toc49681900)

[1.6 ALTER PROCEDURE 16](#_Toc49681901)

[1.7 ALTER SERVER 16](#_Toc49681902)

[1.8 ALTER TABLE 16](#_Toc49681903)

[1.9 ALTER TABLESPACE 16](#_Toc49681904)

[1.10 ALTER VIEW 16](#_Toc49681905)

[1.11 CREATE DATABASE 16](#_Toc49681906)

[1.12 CREATE EVENT 16](#_Toc49681907)

[1.13 CREATE FUNCTION 16](#_Toc49681908)

[1.14 CREATE INDEX 16](#_Toc49681909)

[1.15 CREATE LOGFILE GROUP 16](#_Toc49681910)

[1.16 CREATE PROCEDURE and CREATE FUNCTIONs 16](#_Toc49681911)

[1.17 CREATE SERVER 16](#_Toc49681912)

[1.18 CREATE TABLE 16](#_Toc49681913)

[1.19 CREATE TABLESPACE 16](#_Toc49681914)

[1.20 CREATE TRIGGER 16](#_Toc49681915)

[1.21 CREATE VIEW 16](#_Toc49681916)

[1.22 DROP DATABASE 16](#_Toc49681917)

[1.23 DROP EVENT 16](#_Toc49681918)

[1.24 DROP FUNCTION 16](#_Toc49681919)

[1.25 DROP INDEX 16](#_Toc49681920)

[1.26 DROP LOGFILE GROUP 16](#_Toc49681921)

[1.27 DROP PROCEDURE and DROP FUNCTIONs 16](#_Toc49681922)

[1.28 DROP SERVER 16](#_Toc49681923)

[1.29 DROP TABLE 16](#_Toc49681924)

[1.30 DROP TABLESPACE 16](#_Toc49681925)

[1.31 DROP TRIGGER 16](#_Toc49681926)

[1.32 DROP VIEW 17](#_Toc49681927)

[1.33 RENAME TABLE 17](#_Toc49681928)

[1.34 TRUNCATE TABLE 17](#_Toc49681929)

[2 Data Manipulation Statements（DML） 17](#_Toc49681930)

[2.1 INSERT 17](#_Toc49681931)

[2.2 UPDATE 17](#_Toc49681932)

[2.3 SELECT Statement 17](#_Toc49681933)

[2.4 Subqueries 17](#_Toc49681934)

[2.5 DELETE 17](#_Toc49681935)

[2.6 CALL 17](#_Toc49681936)

[2.7 DO 17](#_Toc49681937)

[2.8 HANDLER 17](#_Toc49681938)

[2.9 LOAD DATA 17](#_Toc49681939)

[2.10 LOAD XML 17](#_Toc49681940)

[2.11 REPLACE 17](#_Toc49681941)

[3 Transactional and Locking Statements 17](#_Toc49681942)

[3.1 Transctional 17](#_Toc49681943)

[3.2 XA Transactions 17](#_Toc49681944)

[3.3 LOCK TABLE 17](#_Toc49681945)

[4 Database Administration Statements 17](#_Toc49681946)

[4.1 SHOW 17](#_Toc49681947)

[4.2 ALTER USER 18](#_Toc49681948)

[4.3 CREATE USER 18](#_Toc49681949)

[4.4 DROP USER 18](#_Toc49681950)

[4.5 GRANT 18](#_Toc49681951)

[4.6 RENAME USER 18](#_Toc49681952)

[4.7 REVOKE 18](#_Toc49681953)

[4.8 SET PASSWORD 18](#_Toc49681954)

[4.9 ANALYZE TABLE 18](#_Toc49681955)

[4.10 CHECK TABLE 18](#_Toc49681956)

[4.11 CHECKSUM TABLE 18](#_Toc49681957)

[4.12 OPTIMIZE TABLE 18](#_Toc49681958)

[4.13 REPAIR TABLE 18](#_Toc49681959)

[4.14 SET Variable 18](#_Toc49681960)

[4.15 Other 18](#_Toc49681961)

[5 Utility Statements 19](#_Toc49681962)

[5.1 EXPLAIN 19](#_Toc49681963)

[5.2 Other 19](#_Toc49681964)

[6 Replication Statements 19](#_Toc49681965)

[7 Compound Statements 19](#_Toc49681966)

[8 Prepared Statements 19](#_Toc49681967)

# 零 参考资料

《高性能Mysql 第三版》

官方文档：

<https://dev.mysql.com/doc/refman/5.7/en/functions.html>

<https://dev.mysql.com/doc/refman/5.7/en/sql-statements.html>

# Functions and Operators

## 概述

## 运算符

<https://dev.mysql.com/doc/refman/5.7/en/non-typed-operators.html>

### 概述

* + - 1. 操作符 =，用于赋值（assignment）还是比较(comparison),优先级不同
      2. 如果同级别优先级，赋值时从右往左，其他情况从左往右

1. --优先级从上到下依次递减
2. INTERVAL
3. **BINARY**, **COLLATE**
4. !
5. - (unary minus), ~ (unary **bit** inversion)
6. ^
7. \*, /, DIV, %, MOD
8. -, +
9. <<, >>
10. &
11. |
12. = (comparison), <=>, >=, >, <=, <, <>, !=, **IS**, LIKE, REGEXP, IN
13. BETWEEN, CASE, **WHEN**, **THEN**, **ELSE**
14. NOT
15. AND, &&
16. XOR
17. OR, ||
18. = (assignment), :=

|  |  |
| --- | --- |
| 运算符 | 描述 |
| 位运算 | |
| & | 位运算，与 |
| >> | 右移位 |
| << | 左移位 |
| %, MOD | 取模 |
| \* | 乘法 |
| + | 加法 |
| - | 减法 |
| - |  |
| -> | 获取指定JSON column值的指定属性; equivalent to JSON\_EXTRACT(). |
| ->> | 获取指定JSON column值的指定属性; equivalent to JSON\_UNQUOTE(JSON\_EXTRACT()).在5.7.13引入 |
| / | 除法 |
| 逻辑条件运算符 | |
| AND, && | 逻辑，与 |
| OR | 逻辑运算，或 |
| NOT, ! | 否定运算符  默认情况下，！的优先级高于NOT  HIGH\_NOT\_PRECEDENCE启用时，两则优先级相同 |
| || | 默认情况下，位逻辑运算符或，同 OR运算符；  当PIPES\_AS\_CONCAT启用时，则为字符串连接符 |
| XOR | 逻辑运算，非或 |
| 其他 | |
| := | 赋值 |
| = | 赋值，用于SET语句 |
| = | 等于 |
| ^ | 位运算，异或 |
| BINARY | 将字符串转成二进制字符串 |
| CASE | Case |
| DIV | 整数除法 |
| REGEXP | 正则 |
| NOT REGEXP | 不匹配正则 |
| RLIKE | Whether string matches regular expression |
| SOUNDS LIKE | Compare sounds |
| | | 位运算，或 |
| ~ | 位运算，取反 |

### 比较运算符

* + - 1. 比较类运算符，结果有1(TRUE)，0(FALSE)，NULL，除了LEAST() 和 GREATEST()
      2. 会根据实际值，数值和字符串会自动转换
      3. 比较符 =,>,<,>=,<=,<>,!= 不仅能用于值比较，还可以用于row comparisons
      4. 默认情况下，字符串比较大小写不敏感

|  |  |
| --- | --- |
| 运算符 | 描述 |
| > | 1. 大于 2. 支持row comparisons即：   (a, b) > (x, y)  等价于  (a > x) OR ((a = x) AND (b > y)) |
| >= | 1. 大于等于    1. 支持row comparisons即：   (a, b) >= (x, y)  等价于  (a > x) OR ((a = x) AND (b >= y)) |
| < | 小于  支持row comparisons即：  (a, b) < (x, y)  等价于  (a < x) OR ((a = x) AND (b < y)) |
| <>, != | 不等于  支持row comparisons即：  (a, b) <> (x, y) 或者(a, b) != (x, y)  等价于  (a <> x) OR (b <> y) |
| <= | 小于等于  支持row comparisons即：  (a, b) <= (x, y)  等价于  (a < x) OR ((a = x) AND (b <= y)) |
| <=> | NULL-safe equal.  具体效果如下   1. **SELECT** 1 <=> 1, NULL <=> NULL, 1 <=> NULL; 2. -> 1, 1, 0 3. **SELECT** 1 = 1, NULL = NULL, 1 = NULL; 4. -> 1, NULL, NULL   支持row comparisons即：  (a, b) <=> (x, y)  等价于  (a <=> x) AND (b <=> y) |
| = | 等于 |
| ***expr*** BETWEEN ***A*** AND ***B*** | 大于等于A，小于等于B之间的值, |
| ***expr*** NOT BETWEEN ***A*** AND ***B*** | 不在大于等于A，小于等于B之间的值 |
| ***expr*** IN() | 1. 值是否在指定集合内，实际判断，通过二进制值判断 2. IN的元素个数由max\_allowed\_packet参数限制，默认4MB 3. 比较结果返回NULL情况   a ***expr***为 NULL  b IN中元素没有匹配值，且其中一个元素为NULL   1. 支持row comparisons即 2. **SELECT** (3,4) IN ((1,2), (3,4)); 3. -> 1 4. **SELECT** (3,4) IN ((1,2), (3,5)); 5. -> 0   5.隐式转换的坑：   1. --这里字符会被转为浮点型0.0,所以结果都为1(TRUE) 2. **SELECT** 'a' IN (0), 0 IN ('b'); 3. -> 1, 1 |
| ***expr*** NOT IN() | 不在指定集合，其他同IN |
| ***A*** IS ***B*** | 判断值是否符合，B的结果值可以是1(TRUE)，0(FALSE)，UNKNOWN   1. **SELECT** 1 **IS** **TRUE**, 0 **IS** **FALSE**, NULL **IS** UNKNOWN; 2. -> 1, 1, 1 |
| IS NULL | 1判断值是否NULL  2对于自增字段：  2.1 如果sql\_auto\_is\_null参数为1，则下面语句会返回最后插入的记录  **SELECT** \* **FROM** tbl\_name **WHERE** auto\_col **IS** NULL  2.2 如果sql\_auto\_is\_null参数为0，则不返回数据  3 对于DATE和DATETIME的非空字段，对应的0值数据，可以通过下面方式查找：  SELECT \* FROM tbl\_name WHERE date\_column IS NULL |
| IS NOT | 判断值是否不符合，其他与IS 相同 |
| IS NOT NULL | 判断值是否非NULL，其他与IS NULL相同 |
| LIKE | 简单的模式匹配 |
| NOT LIKE | 不符合简单的模式匹配 |
| COALESCE(...) | 返回集合中第一个非NULL值，如果集合中都是NULL，则返回NULL |
| GREATEST(...) | 返回集合中最大的值；  如果是字符串，则按照指定的字符集排序规则获取最大值；  如果集合中都是NULL，则返回NULL |
| INTERVAL(N，N1，N2...) | 通过二进制比较，在N1，N2，N3。。。集合中找到第一个大于N的值下标  下标从0开始，即如果N1>N，则返回0，如果N2>N返回1  如果N为NULL或找不到，返回-1 |
| ISNULL(expr) | 如果expr结果值异常或为NULL，则返回1，其他情况返回0   1. **SELECT** ISNULL(1+1); 2. -> 0 3. **SELECT** ISNULL(1/0); 4. -> 1   因为值与NULL比较，结果总为NULL，所以，可以用ISNULL代替  其他大部分行为跟IS NULL 相同 |
| LEAST(v1,v2,v3...) | 获取集合中最小的值  如果集合元素都是NULL，返回NULL  如果元素都是整数，按整数比较  如果元素有一个是双精度浮点型，都转为转双精度浮点型比较  如果元素有一个是DECIMAL，都转为转DECIMAL比较  如果元素是数值和字符混合，则都按数值比较  如果元素都是非二进制字符串，则都按非二进制字符串比较  如果其他情况，都按对应的二进制字符串比较 |

## Control Flow Functions

|  |  |
| --- | --- |
| **Name** | **Description** |
| CASE | CASE [value] WHEN [condition | compareValue] THEN result [WHEN [condition | compareValue] THEN result ...] [ELSE result] END  1 当指定value时，返回符合第一个WHEN CompareValue 对应的result，如果都不符合WHEN，则如果有指定ELSE result 则返回ELSE指定的result，否则返回NULL  2 当没有指定value时，返回符合第一个WHEN condition对应的result，如果都不符合WHEN，则如果有指定ELSE result 则返回ELSE指定的result，否则返回NULL |
| IF() | IF(expr1,expr2,expr3) 等同于 expr1?expr2:expr3  依次遵循以下规则  1 只有expr2或expr3显示指定NULL时，才有可能返回NULL  2 如果expr2和expr3有一个是String类型，返回值都是String  3 如果expr2和expr3都是String类型其中一个是大小写敏感的，则返回值都是大小写敏感的string  4 如果expr2和expr3有一个是floating-point类型，返回值都是floating-point  5 如果expr2和expr3有一个是整数，返回值都是整数 |
| IFNULL() | IFNULL(expr1,expr2) 等同于expr1 !=NULL ？expr1: expr2  实际返回类型，按兼容expr1和expr2的通用类型返回，依次为string，real，integer |
| NULLIF() | NULLIF(expr1,expr2) 等同于expr1==expr2？NULL:expr1  返回值类型同expr1 |

## 数字相关 Functions & Operator

<https://dev.mysql.com/doc/refman/5.7/en/numeric-functions.html>

### 计算运算符

* + - 1. 计算结果遵循以下规则
         1. 如果两个数字都是integer，对于 - + \* 运算符，结果均为BIGINT(64 bit)类型
         2. 如果两个数字都是integer且其中一个是unsigned，则结果为unsigned integer
         3. 如果SQL mode 是NO\_UNSIGNED\_SUBTRACTION，则对于减法，无论两个整数数字是否unsigned，其结果都是signed
         4. 对于 +, -, /, \*, % 运算符，两个数字其中一个是real或string类型，则结果的精度为能支持结果的最大精度
         5. 对于 / 运算符，结果的小数点长度为，第一个计算数的小数点位数 加上 系统参数div\_precision\_increment的值(默认为4)，例子：5.05 / 0.014 = 360.714286
         6. a.5的规则适用于计算过程中每一步，例子(14620/9432456)/(24250/9432456)=(0.0014)/(0.0026)= 0.60288653

|  |  |
| --- | --- |
| 运算符 | 描述 |
| %, MOD | 取模，等同 MOD() 方法 |
| \* | 乘法 |
| + | 加法 |
| - | 减法 |
| - | 负号，尽量避免对BIGINT使用 |
| / | 除法  1 除0时，返回NULL  2 仅当需要强转为integer时，才用BIGINT类型进行计算 |
| DIV | 整数除法  1 如果两个数至少有一个非integer，则将两个数字转为Decimal类型进行计算，并将结果转成BIGINT  2 如果结果值超过了BIGINT范围，将报错 |

### Function

|  |  |
| --- | --- |
| Name | **Description** |
| ABS() | ABS(X)  返回绝对值，返回值类型为BIGINT |
| ACOS() | ACOS(X)  1 反余弦函数  2 如果x非-1 to 1，返回NULL |
| ASIN() | ASIN(X)  1反正弦函数  2 如果x非-1 to 1，返回NULL |
| ATAN() | ATAN (X) |
| ATAN2()  ATAN() | ATAN(Y,X), ATAN2(Y,X) |
| CEIL() | CEIL(X),同CEILING() |
| CEILING() | CEILING(X)  1 返回大于X的最小整数  2 如果X为字符串或floating-point类型，则返回值类型为floating-point |
| CONV() | CONV(N,from\_base,to\_base)  1 将N值从from\_base转到to\_base进制  2 基于64bit计算  3 如果任一个参数为NULL，返回NULL  4 from\_base/to\_base的最小值为2，最大值为36   1. SELECT CONV('a',16,2); 2. -> '1010' 3. SELECT CONV('6E',18,8); 4. -> '172' 5. SELECT CONV(-17,10,-18); 6. -> '-H' 7. SELECT CONV(10+'10'+'10'+X'0a',10,10); 8. -> '40' |
| COS() | COS(X)  余弦函数 |
| COT() | COT(X)  余切函数 |
| CRC32() | CRC32(expr)  1 获取expr的CRC32码  2 如果expr为NULL，则返回NULL  3 expr值必须是String，如果不是，则会被转为String进行处理 |
| DEGREES() | DEGREES(X)  将弧度X转为角度   1. SELECT DEGREES(PI()); 2. -> 180 |
| EXP() | EXP(X)  返回e的X次方值   1. SELECT EXP(2); 2. -> 7.3890560989307 |
| FLOOR() | FLOOR(X)  返回小于X的最大整数  1 如果X是字符串或floating-point类型，返回值为floating-point类型  2 其他情况返回值类型同X类型 |
| LN() | LN(X)  1返回X的自然对数  2 如果X小于等于0，则返回NULL，并抛出WARNING |
| LOG() | LOG(X)  同EXP(X)  LOG(B,X)  1 等同于log**B**X  2 如果B小于等于1，返回NULL   1. **SELECT** LOG(2,65536); 2. -> 16 |
| LOG10() | LOG10(X)  log**10**X  如果X小于等于0，返回NULL并抛出WARNING |
| LOG2() | LOG2(X)  log**2**X  如果X小于等于0，返回NULL并抛出WARNING |
| MOD() | MOD(N,M)  等同N%M  M为0时，返回NULL |
| PI() | 返回π值，默认返回7位小数   1. **SELECT** PI(); 2. -> 3.141593 3. **SELECT** PI()+0.000000000000000000; 4. -> 3.141592653589793116 |
| POW() | POW(X,Y)  即X**Y** |
| POWER() | 同POW() |
| RADIANS() | RADIANS(X)  将角度X转为弧度 |
| RAND() | RAND([N])  1 如果未指定N，则随机返回一个floating-point v值，0<=v<1  2 如果指定N，N将被作为种子值 |
| ROUND() | 1如果X值为精准数字，则结果为四舍五入(“round half away from zero” or “round toward nearest”)  2如果X值为近似值，则依赖当前系统采用的C语言，取最近的偶数(round to nearest even)   1. **SELECT** ROUND(2.5), ROUND(25E-1); 2. +------------+--------------+ 3. | ROUND(2.5) | ROUND(25E-1) | 4. +------------+--------------+ 5. | 3          |            2 | 6. +------------+--------------+   **ROUND(X)**  1等同于ROUND(X,0)  2返回值类型跟X相同  **ROUND(X,D)**   1. 返回值类型跟X相同 2. D如果没有指定，默认为0 3. D如果指定则保留指定位数的小数部分 4. D最大值为30，最小值为-30，如果超出范围，则超出部分被截取 5. **SELECT** ROUND(-1.23); 6. -> -1 7. **SELECT** ROUND(-1.58); 8. -> -2 9. **SELECT** ROUND(1.58); 10. -> 2 11. **SELECT** ROUND(1.298, 1); 12. -> 1.3 13. **SELECT** ROUND(1.298, 0); 14. -> 1 15. **SELECT** ROUND(23.298, -1); 16. -> 20 |
| SIGN() | SIGN(X)  1 判断X为正负数  2 X=负数返回 -1，X=0返回0，X=正数返回1 |
| SIN() | SIN(X)  返回弧度为X的sin值 |
| SQRT() | SQRT(X)  1 返回非负数的X的平方根  2 如果X为负数，则返回NULL |
| TAN() | TAN(X)  返回弧度为X的tan值 |
| TRUNCATE() | TRUNCATE(X,D)  1 如果D>0 ，保留小数点后面D位小数  2 如果D=0，则去掉小数点后面数值  3 如果D<0,则从小数点开始，从右往左用0替代D位数值   1. **SELECT** **TRUNCATE**(1.223,1); 2. -> 1.2 3. **SELECT** **TRUNCATE**(1.999,1); 4. -> 1.9 5. **SELECT** **TRUNCATE**(1.999,0); 6. -> 1 7. **SELECT** **TRUNCATE**(-1.999,1); 8. -> -1.9 9. **SELECT** **TRUNCATE**(122,-2); 10. -> 100 11. **SELECT** **TRUNCATE**(10.28\*100,0); 12. -> 1028 |

## Date and Time function

<https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html>

### 如NOW()、CURDATE()、CURTIME()、UTC\_DATE()、UTC\_TIME()、UTC\_TIMESTAMP()以及它们的同义词，每次query请求，每相同的Function只执行一次，且以第一次执行结果为准，即如果有多个NOW()，则只以第一个NOW()计算结果为准

### 如CURRENT\_TIMESTAMP()、CURRENT\_TIME()、CURRENT\_DATE()、FROM\_UNIXTIME()返回值依赖当前session所处时区。

### 时区相关：

<https://dev.mysql.com/doc/refman/5.7/en/time-zone-support.html>

### 具体函数

|  |  |
| --- | --- |
| Name | **Description** |
| ADDDATE() | 同义词：DATE\_ADD()  ADDDATE(date,INTERVAL expr UNIT)  1、将date加上指定单位量  2、UNIT取值：  <https://dev.mysql.com/doc/refman/5.7/en/expressions.html#temporal-intervals>   1. **SELECT** ADDDATE('2008-01-02', INTERVAL 31 DAY); 2. -> '2008-02-02'   ADDDATE(expr,days)  1、将expr 加上days天数  2、等同ADDDATE('2008-01-02', INTERVAL 31 DAY)   1. **SELECT** ADDDATE('2008-01-02', 31); 2. -> '2008-02-02' |
| ADDTIME() | ADDTIME(expr1,expr2)  将时间或日期(expr1),加上时间(expr2)   1. **SELECT** ADDTIME('2007-12-31 23:59:59.999999', '1 1:1:1.000002'); 2. -> '2008-01-02 01:01:01.000001' 3. **SELECT** ADDTIME('01:00:00.999999', '02:00:00.999998'); 4. -> '03:00:01.999997' |
| CONVERT\_TZ() | CONVERT\_TZ(dt,from\_tz,to\_tz)  1 将日期dt，从from\_tz 时区，转成to\_tz时区的时间  2 如果参数不合法，将返回NULL  3 如果to\_tz为UTC，且结果值超过了UTC范围，则不发生转换操作 |
| CURDATE() | CURDATE()  返回YYYY-MM-DD 或YYYYMMDD格式的当前日期   1. **SELECT** CURDATE(); 2. -> '2008-06-13' 3. **SELECT** CURDATE() + 0; 4. -> 20080613 |
| CURRENT\_DATE()  CURRENT\_DATE | 同义词：CURDATE() |
| CURRENT\_TIME()  CURRENT\_TIME | 同义词：CURTIME()  CURRENT\_TIME, CURRENT\_TIME([fsp]) |
| CURRENT\_TIMESTAMP()  CURRENT\_TIMESTAMP | 同义词 NOW()  CURRENT\_TIMESTAMP, CURRENT\_TIMESTAMP([fsp]) |
| CURTIME() | CURTIME([fsp])  1 返回'hh:mm:ss' 或 hhmmss的当前时间  2 通过fsp指定时间小数点位数，默认0，取值范围：0-6   1. **SELECT** CURTIME(); 2. -> '23:50:26' 3. **SELECT** CURTIME() + 0; 4. -> 235026.000000 |
| DATE() | DATE(expr)  获取指定日期时间expr中的日期   1. **SELECT** **DATE**('2003-12-31 01:02:03'); 2. -> '2003-12-31' |
| DATE\_ADD() | DATE\_ADD(date,INTERVAL expr unit)  1 将指定日期/日期时间，加上指定单位unit的expr量  2 unit取值  <https://dev.mysql.com/doc/refman/5.7/en/expressions.html#temporal-intervals>  3 expr实际计算时按string处理  4 如果expr为负数，则以相反计算，比如加 -> 减，减 -> 加  5 如果date是  6 unit 取值参考5.6 |
| DATE\_FORMAT() | DATE\_FORMAT(date,format)  将date日期，格式化为format样式  返回字符串，可以包含非ASCII字符  format样式占位符参考 5.5   1. **SELECT** DATE\_FORMAT('2009-10-04 22:23:00', '%W %M %Y'); 2. -> 'Sunday October 2009' 3. **SELECT** DATE\_FORMAT('2007-10-04 22:23:00', '%H:%i:%s'); 4. -> '22:23:00' 5. **SELECT** DATE\_FORMAT('1900-10-04 22:23:00', '%D %y %a %d %m %b %j'); 6. -> '4th 00 Thu 04 10 Oct 277' 7. **SELECT** DATE\_FORMAT('1997-10-04 22:23:00', '%H %k %I %r %T %S %w'); 8. -> '22 22 10 10:23:00 PM 22:23:00 00 6' 9. **SELECT** DATE\_FORMAT('1999-01-01', '%X %V'); 10. -> '1998 52' 11. **SELECT** DATE\_FORMAT('2006-06-00', '%d'); 12. -> '00' |
| DATE\_SUB() | DATE\_SUB(date,INTERVAL expr unit)  1将指定日期/日期时间，减去指定单位unit的expr量   1. 其他特性同DATE\_ADD 2. unit 取值参考5.6 |
| DATEDIFF() | DATEDIFF(expr1,expr2)  1计算日期时间expr1 – 日期时间expr2的天数  2 即使expr1或expr2包含时间，只计算日期部分 |
| DAY() | 同DAYOFMONTH() |
| DAYNAME() | DAYNAME(date)  返回指定日期的星期几的本地语言名，具体根据系统参数lc\_time\_names确定   1. **SELECT** DAYNAME('2007-02-03'); 2. -> 'Saturday' 3. **SET** lc\_time\_names = 'es\_MX'; 4. **SELECT** DAYNAME('2010-01-01'); 5. -> viernes |
| DAYOFMONTH() | DAYOFMONTH(date)  返回指定日期的日，数字表示(0-31) |
| DAYOFWEEK() | DAYOFWEEK(date)  返回指定日期是星期几(1 = Sunday, 2 = Monday, …, 7 = Saturday)  符合ODBC标准 |
| DAYOFYEAR() | DAYOFYEAR(date)  返回指定日期是当年第几天(1-366) |
| EXTRACT() | EXTRACT(unit FROM date)  从date中提取指定数据(unit)  unit 取值参考5.6   1. mysql> **SELECT** EXTRACT(YEAR **FROM** '2019-07-02'); 2. -> 2019 3. mysql> **SELECT** EXTRACT(YEAR\_MONTH **FROM** '2019-07-02 01:02:03'); 4. -> 201907 5. mysql> **SELECT** EXTRACT(DAY\_MINUTE **FROM** '2019-07-02 01:02:03'); 6. -> 20102 7. mysql> **SELECT** EXTRACT(MICROSECOND **FROM** '2003-01-02 10:30:00.000123'); 8. -> 123 |
| FROM\_DAYS() | FROM\_DAYS(N)   1. **SELECT** FROM\_DAYS(730669); 2. -> '2000-07-03' |
| FROM\_UNIXTIME() | FROM\_UNIXTIME(unix\_timestamp[,format])  将时间戳unix\_timestamp按照format格式展示  format 默认为YYYY-MM-DD hh:mm:ss 格式，占位符参考5.5 |
| GET\_FORMAT() | GET\_FORMAT({DATE|TIME|DATETIME},{'EUR'|'USA'|'JIS'|'ISO'|'INTERNAL'})  获取指定默认日期格式，具体参考5.7   1. **SELECT** DATE\_FORMAT('2003-10-03',GET\_FORMAT(**DATE**,'EUR')); 2. -> '03.10.2003' 3. **SELECT** STR\_TO\_DATE('10.31.2003',GET\_FORMAT(**DATE**,'USA')); 4. -> '2003-10-31' |
| HOUR() | HOUR(time)  返回指定时间的小时  小时值，实际可以大于23   1. **SELECT** **HOUR**('10:05:03'); 2. -> 10 3. **SELECT** **HOUR**('272:59:59'); 4. -> 272 |
| LAST\_DAY | LAST\_DAY(date)  返回指定日期的当月最后一天   1. **SELECT** LAST\_DAY('2003-02-05'); 2. -> '2003-02-28' 3. **SELECT** LAST\_DAY('2004-02-05'); 4. -> '2004-02-29' 5. **SELECT** LAST\_DAY('2004-01-01 01:01:01'); 6. -> '2004-01-31' 7. **SELECT** LAST\_DAY('2003-03-32'); 8. -> NULL |
| LOCALTIME()  LOCALTIME | Synonym for NOW() |
| LOCALTIMESTAMP  LOCALTIMESTAMP() | Synonym for NOW() |
| MAKEDATE() | MAKEDATE(year,dayofyear)  获取指定年份year的第dayofyear天的日期   1. **SELECT** MAKEDATE(2011,31), MAKEDATE(2011,32); 2. -> '2011-01-31', '2011-02-01' 3. **SELECT** MAKEDATE(2011,365), MAKEDATE(2014,365); 4. -> '2011-12-31', '2014-12-31' 5. **SELECT** MAKEDATE(2011,0); 6. -> NULL |
| MAKETIME() | MAKETIME(hour,minute,second)  生成时分秒时间，second可以有小数部分  SELECT MAKETIME(12,15,30);  -> '12:15:30' |
| 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 |

### 日期时间 FORMAT 占位符

|  |  |
| --- | --- |
| 占位符 | 描述 |
| %a | 星期英文缩写(Sun..Sat) |
| %b | 月份 英文缩写 (Jan..Dec) |
| %c | 月份，数字表示 (0..12) |
| %D | 日，英文后缀表示 (0th, 1st, 2nd, 3rd, …) |
| %d | 日, 双位数字表示 (00..31) |
| %e | 日, 数字表示 (0..31) |
| %f | 微秒 (000000..999999) |
| %H | 小时（24小时），双数字表示(00..23) |
| %h | 小时（12小时），双数字表示 (01..12) |
| %I | 同%H |
| %i | 分, 双数字表示 (00..59) |
| %j | 年中第几天，三数字表示 (001..366) |
| %k | 小时（24小时），数字表示(0..23) |
| %l | 小时（12小时），数字表示(1..12) |
| %M | 月，英文名 (January..December) |
| %m | 月, 双数字表示 (00..12) |
| %p | AM or PM |
| %r | 当前时间, 12-hour (*hh:mm:ss* followed by AM or PM) |
| %S | 秒，双数字表示 (00..59) |
| %s | 同%S |
| %T | 当前时间, 24-hour (*hh:mm:ss*) |
| %U | 获取当前是今年第几周(00..53)，星期日是一周的第一天; [WEEK()](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_week) mode 0 |
| %u | 获取当前是今年第几周(00..53)，星期一是一周的第一天； [WEEK()](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_week) mode 1 |
| %V | 获取当前是今年第几周(01..53), 星期日是一周的第一天; [WEEK()](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_week) mode 2; 跟 %X 配合 |
| %v | 获取当前是今年第几周(01..53), 星期一是一周的第一天; [WEEK()](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_week) mode 3; 跟 %x 配合使用 |
| %W | 星期几英文名 (Sunday..Saturday) |
| %w | 星期几，数字表示 (0=Sunday..6=Saturday) |
| %X | 年，星期日是一周的第一天，四位数; 跟%V 配合使用  SELECT DATE\_FORMAT("2019-01-01", "%X%V") AS currentWeek;  -> 201852 |
| %x | 年，星期一是一周的第一天，四位数，跟%v 配合使用 |
| %Y | 自然年, 四位数字 |
| %y | 自然年, 两位数字 |
| %% | % 字符 |
| %*x* | *x*, for any “*x*” not listed above |

### 时间单位

|  |  |
| --- | --- |
| ***unit* Value** | **Expected *expr* Format** |
| MICROSECOND | MICROSECONDS |
| SECOND | SECONDS |
| MINUTE | MINUTES |
| HOUR | HOURS |
| DAY | DAYS |
| WEEK | WEEKS |
| MONTH | MONTHS |
| QUARTER | QUARTERS |
| YEAR | YEARS |
| SECOND\_MICROSECOND | 'SECONDS.MICROSECONDS' |
| MINUTE\_MICROSECOND | 'MINUTES:SECONDS.MICROSECONDS' |
| MINUTE\_SECOND | 'MINUTES:SECONDS' |
| HOUR\_MICROSECOND | 'HOURS:MINUTES:SECONDS.MICROSECONDS' |
| HOUR\_SECOND | 'HOURS:MINUTES:SECONDS' |
| HOUR\_MINUTE | 'HOURS:MINUTES' |
| DAY\_MICROSECOND | 'DAYS HOURS:MINUTES:SECONDS.MICROSECONDS' |
| DAY\_SECOND | 'DAYS HOURS:MINUTES:SECONDS' |
| DAY\_MINUTE | 'DAYS HOURS:MINUTES' |
| DAY\_HOUR | 'DAYS HOURS' |
| YEAR\_MONTH | 'YEARS-MONTHS' |

### GET\_FORMAT

|  |  |
| --- | --- |
| **Function Call** | **Result** |
| [GET\_FORMAT(DATE,'USA')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%m.%d.%Y' |
| [GET\_FORMAT(DATE,'JIS')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%Y-%m-%d' |
| [GET\_FORMAT(DATE,'ISO')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%Y-%m-%d' |
| [GET\_FORMAT(DATE,'EUR')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%d.%m.%Y' |
| [GET\_FORMAT(DATE,'INTERNAL')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%Y%m%d' |
| [GET\_FORMAT(DATETIME,'USA')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%Y-%m-%d %H.%i.%s' |
| [GET\_FORMAT(DATETIME,'JIS')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%Y-%m-%d %H:%i:%s' |
| [GET\_FORMAT(DATETIME,'ISO')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%Y-%m-%d %H:%i:%s' |
| [GET\_FORMAT(DATETIME,'EUR')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%Y-%m-%d %H.%i.%s' |
| [GET\_FORMAT(DATETIME,'INTERNAL')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%Y%m%d%H%i%s' |
| [GET\_FORMAT(TIME,'USA')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%h:%i:%s %p' |
| [GET\_FORMAT(TIME,'JIS')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%H:%i:%s' |
| [GET\_FORMAT(TIME,'ISO')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%H:%i:%s' |
| [GET\_FORMAT(TIME,'EUR')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%H.%i.%s' |
| [GET\_FORMAT(TIME,'INTERNAL')](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_get-format) | '%H%i%s' |

## String Function and operators

|  |  |
| --- | --- |
| 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() | Index (position) of first argument in subsequent arguments |
| FIND\_IN\_SET() | Index (position) of first argument within second argument |
| FORMAT() | Return a number formatted to specified number of decimal places |
| FROM\_BASE64() | Decode base64 encoded string and return result |
| HEX() | Hexadecimal representation of decimal or string value |
| INSERT() | Insert substring at specified position up to 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 | Whether string matches regular expression |
| 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 | Whether string matches regular expression |
| 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 |

## Aggregate Functions

|  |  |
| --- | --- |
| Name | Description |
| AVG() | Return the average value of the argument |
| BIT\_AND() | Return bitwise AND |
| BIT\_OR() | Return bitwise OR |
| BIT\_XOR() | Return bitwise XOR |
| COUNT() | Return a count of the number of rows returned |
| COUNT(DISTINCT) | Return the count of a number of different values |
| GROUP\_CONCAT() | Return a concatenated string |
| JSON\_ARRAYAGG() | Return result set as a single JSON array |
| JSON\_OBJECTAGG() | Return result set as a single JSON object |
| MAX() | Return the maximum value |
| MIN() | Return the minimum value |
| STD() | Return the population standard deviation |
| STDDEV() | Return the population standard deviation |
| STDDEV\_POP() | Return the population standard deviation |
| STDDEV\_SAMP() | Return the sample standard deviation |
| SUM() | Return the sum |
| VAR\_POP() | Return the population standard variance |
| VAR\_SAMP() | Return the sample variance |
| VARIANCE() | Return the population standard variance |

## Other 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 |
| ExtractValue() | Extract a value from an XML string using XPath notation |
| UpdateXML() | Return replaced XML fragment |
| & | Bitwise AND |
| >> | Right shift |
| << | Left shift |
| ^ | Bitwise XOR |
| BIT\_COUNT() | Return the number of bits that are set |
| | | Bitwise OR |
| ~ | Bitwise inversion |
| AES\_DECRYPT() | Decrypt using AES |
| AES\_ENCRYPT() | Encrypt using AES |
| ASYMMETRIC\_DECRYPT() | Decrypt ciphertext using private or public key |
| ASYMMETRIC\_DERIVE() | Derive symmetric key from asymmetric keys |
| ASYMMETRIC\_ENCRYPT() | Encrypt cleartext using private or public key |
| ASYMMETRIC\_SIGN() | Generate signature from digest |
| ASYMMETRIC\_VERIFY() | Verify that signature matches digest |
| COMPRESS() | Return result as a binary string |
| CREATE\_ASYMMETRIC\_PRIV\_KEY() | Create private key |
| CREATE\_ASYMMETRIC\_PUB\_KEY() | Create public key |
| CREATE\_DH\_PARAMETERS() | Generate shared DH secret |
| CREATE\_DIGEST() | Generate digest from string |
| DECODE() (deprecated) | Decode a string encrypted using ENCODE() |
| DES\_DECRYPT() (deprecated) | Decrypt a string |
| DES\_ENCRYPT() (deprecated) | Encrypt a string |
| ENCODE() (deprecated) | Encode a string |
| ENCRYPT() (deprecated) | Encrypt a string |
| MD5() | Calculate MD5 checksum |
| PASSWORD() (deprecated) | Calculate and return a password string |
| RANDOM\_BYTES() | Return a random byte vector |
| SHA1(), SHA() | Calculate an SHA-1 160-bit checksum |
| SHA2() | Calculate an SHA-2 checksum |
| UNCOMPRESS() | Uncompress a string compressed |
| UNCOMPRESSED\_LENGTH() | Return the length of a string before compression |
| VALIDATE\_PASSWORD\_STRENGTH() | Determine strength of password |
| GET\_LOCK() | Get a named lock |
| IS\_FREE\_LOCK() | Whether the named lock is free |
| IS\_USED\_LOCK() | Whether the named lock is in use; return connection identifier if true |
| RELEASE\_ALL\_LOCKS() | Release all current named locks |
| RELEASE\_LOCK() | Release the named lock |
| BENCHMARK() | Repeatedly execute an expression |
| CHARSET() | Return the character set of the argument |
| COERCIBILITY() | Return the collation coercibility value of the string argument |
| COLLATION() | Return the collation of the string argument |
| CONNECTION\_ID() | Return the connection ID (thread ID) for the connection |
| CURRENT\_USER(), CURRENT\_USER | The authenticated user name and host name |
| DATABASE() | Return the default (current) database name |
| FOUND\_ROWS() | For a SELECT with a LIMIT clause, the number of rows that would be returned were there no LIMIT clause |
| LAST\_INSERT\_ID() | Value of the AUTOINCREMENT column for the last INSERT |
| ROW\_COUNT() | The number of rows updated |
| SCHEMA() | Synonym for DATABASE() |
| SESSION\_USER() | Synonym for USER() |
| SYSTEM\_USER() | Synonym for USER() |
| USER() | The user name and host name provided by the client |
| VERSION() | Return a string that indicates the MySQL server version |
| ANY\_VALUE() | Suppress ONLY\_FULL\_GROUP\_BY value rejection |
| DEFAULT() | Return the default value for a table column |
| INET\_ATON() | Return the numeric value of an IP address |
| INET\_NTOA() | Return the IP address from a numeric value |
| INET6\_ATON() | Return the numeric value of an IPv6 address |
| INET6\_NTOA() | Return the IPv6 address from a numeric value |
| IS\_IPV4() | Whether argument is an IPv4 address |
| IS\_IPV4\_COMPAT() | Whether argument is an IPv4-compatible address |
| IS\_IPV4\_MAPPED() | Whether argument is an IPv4-mapped address |
| IS\_IPV6() | Whether argument is an IPv6 address |
| MASTER\_POS\_WAIT() | Block until the slave has read and applied all updates up to the specified position |
| NAME\_CONST() | Cause the column to have the given name |
| SLEEP() | Sleep for a number of seconds |
| UUID() | Return a Universal Unique Identifier (UUID) |
| UUID\_SHORT() | Return an integer-valued universal identifier |
| VALUES() | Define the values to be used during an INSERT |

# SQL Statements

## Data Definition Statements（DDL）

### ALTER DATABASE

### ALTER EVENT

### ALTER FUNCTION

### ALTER INSTANCE

### ALTER LOGFILE GROUP

### ALTER PROCEDURE

### ALTER SERVER

### ALTER TABLE

### ALTER TABLESPACE

### ALTER VIEW

### CREATE DATABASE

### CREATE EVENT

### CREATE FUNCTION

### CREATE INDEX

### CREATE LOGFILE GROUP

### CREATE PROCEDURE and CREATE FUNCTIONs

### CREATE SERVER

### CREATE TABLE

### CREATE TABLESPACE

### CREATE TRIGGER

### CREATE VIEW

### DROP DATABASE

### DROP EVENT

### DROP FUNCTION

### DROP INDEX

### DROP LOGFILE GROUP

### DROP PROCEDURE and DROP FUNCTIONs

### DROP SERVER

### DROP TABLE

### DROP TABLESPACE

### DROP TRIGGER

### DROP VIEW

### RENAME TABLE

### TRUNCATE TABLE

## Data Manipulation Statements（DML）

### INSERT

### UPDATE

### SELECT Statement

### Subqueries

### DELETE

### CALL

### DO

### HANDLER

### LOAD DATA

### LOAD XML

### REPLACE

## Transactional and Locking Statements

### Transctional

### XA Transactions

### LOCK TABLE

## Database Administration Statements

### SHOW

|  |  |
| --- | --- |
| 语句 | 说明 |
| SHOW BINARY LOGS |  |
| SHOW BINLOG EVENTS |  |
| SHOW CHARACTER SET |  |
| SHOW COLLATION |  |
| SHOW COLUMNS |  |
| SHOW CREATE DATABASE |  |
| SHOW CREATE EVENT |  |
| SHOW CREATE FUNCTION |  |
| SHOW CREATE PROCEDURE |  |
| SHOW CREATE TABLE |  |
| SHOW CREATE TRIGGER |  |
| SHOW CREATE USER |  |
| SHOW CREATE VIEW |  |
| SHOW DATABASES |  |
| SHOW ENGINE |  |
| SHOW ENGINES |  |
| SHOW ERRORS |  |
| SHOW EVENTS |  |
| SHOW FUNCTION CODE |  |
| SHOW FUNCTION STATUS |  |
| SHOW GRANTS |  |
| SHOW INDEX |  |
| SHOW MASTER STATUS |  |
| SHOW OPEN TABLES |  |
| SHOW PLUGINS |  |
| SHOW PRIVILEGES |  |
| SHOW PROCEDURE CODE |  |
| SHOW PROCEDURE STATUS |  |
| SHOW PROCESSLIST |  |
| SHOW PROFILE |  |
| SHOW PROFILES |  |
| SHOW RELAYLOG EVENTS |  |
| SHOW SLAVE HOSTS |  |
| SHOW SLAVE STATUS |  |
| SHOW STATUS |  |
| SHOW TABLE STATUS |  |
| SHOW TABLES |  |
| SHOW TRIGGERS |  |
| SHOW VARIABLES |  |
| SHOW WARNINGS |  |

### ALTER USER

### CREATE USER

### DROP USER

### GRANT

### RENAME USER

### REVOKE

### SET PASSWORD

### ANALYZE TABLE

### CHECK TABLE

### CHECKSUM TABLE

### OPTIMIZE TABLE

### REPAIR TABLE

### SET Variable

### Other

|  |  |
| --- | --- |
| 语句 | 说明 |
| CREATE FUNCTION |  |
| DROP FUNCTION |  |
| INSTALL PLUGIN |  |
| UNINSTALL PLUGIN |  |
| BINLOG |  |
| CACHE INDEX |  |
| FLUSH |  |

## Utility Statements

### EXPLAIN

### Other

|  |  |
| --- | --- |
| 描述 | 说明 |
| HELP |  |
| USE |  |

## Replication Statements

## Compound Statements

## Prepared Statements