mysql 函数大全

对于针对字符串位置的操作，第一个位置被标记为 1。

ASCII(str)

返回字符串 str 的最左面字符的 ASCII 代码值。如果 str 是空字符串，返回 0。如果 str 是 NULL，返回 NULL。

```
mysql> select ASCII('2');
    -> 50
mysql> select ASCII(2);
    -> 50
mysql> select ASCII('dx');
    -> 100
```

也可参见 ORD()函数。

ORD(str)

如果字符串 str 最左面字符是一个多字节字符，通过以格式((first byte ASCII code)*256+(second byte ASCII code))[*256+third byte ASCII code...]返回字符的 ASCII 代码值来返回多字节字符代码。如果最左面的字符不是一个多字节字符。返回与 ASCII()函数返回的相同值。

```
mysql> select ORD('2');
    -> 50
```

CONV(N,from_base,to_base)

在不同的数字基之间变换数字。返回数字 N 的字符串数字，从 from_base 基变换为 to_base 基，如果任何参数是 NULL，返回 NULL。参数 N 解释为一个整数，但是可以指定为一个整数或一个字符串。最小基是 2 且最大的基是 36。如果 to_base 是一个负数，N 被认为是一个有符号数，否则，N 被当作无符号数。 CONV 以 64 位点精度工作。

```
mysql> select CONV("a",16,2);
    -> '1010'
mysql> select CONV("6E",18,8);
    -> '172'
mysql> select CONV(-17,10,-18);
    -> '-H'
mysql> select CONV(10+"10"+'10'+0xa,10,10);
    -> '40'
```

BIN(N)

返回二进制值 N 的一个字符串表示，在此 N 是一个长整数(BIGINT)数字，这等价于 CONV(N,10,2)。如果 N 是 NULL，返回 NULL。

```
mysql> select BIN(12);
    -> '1100'
```

OCT(N)

返回八进制值 N 的一个字符串的表示，在此 N 是一个长整型数字，这等价于 CONV(N,10,8)。如果 N 是 NULL，返回 NULL。

```
mysql> select OCT(12);
    -> '14'
```

HEX(N)

返回十六进制值 N 一个字符串的表示，在此 N 是一个长整型(BIGINT)数字，这等价于 CONV(N,10,16)。如果 N 是 NULL，返回 NULL。

```
mysql> select HEX(255);
    -> 'FF'
```

CHAR(N,...)

CHAR()将参数解释为整数并且返回由这些整数的 ASCII 代码字符组成的一个字符串。NULL 值被跳过。

```
mysql> select CHAR(77,121,83,81,'76');
    -> 'MySQL'
mysql> select CHAR(77,77.3,'77.3');
    -> 'MMM'
```

CONCAT(str1,str2,...)

返回来自于参数连结的字符串。如果任何参数是 NULL，返回 NULL。可以有超过 2 个的参数。一个数字参数被变换为等价的字符串形式。

```
mysql> select CONCAT('My', 'S', 'QL');
    -> 'MySQL'
mysql> select CONCAT('My', NULL, 'QL');
    -> NULL
mysql> select CONCAT(14.3);
    -> '14.3'
```

LENGTH(str)

OCTET_LENGTH(str)

CHAR_LENGTH(str)

CHARACTER_LENGTH(str)

返回字符串 str 的长度。

```
mysql> select LENGTH('text');
    -> 4
mysql> select OCTET_LENGTH('text');
    -> 4
```

注意，对于多字节字符，其 CHAR_LENGTH()仅计算一次。

LOCATE(substr,str)

POSITION(substr IN str)

返回子串 substr 在字符串 str 第一个出现的位置，如果 substr 不是在 str 里面，返回 0.

```
mysql> select LOCATE('bar', 'foobarbar');
    -> 4
mysql> select LOCATE('xbar', 'foobar');
    -> 0
```

该函数是多字节可靠的。

LOCATE(substr,str,pos)

返回子串 substr 在字符串 str 第一个出现的位置，从位置 pos 开始。如果 substr 不是在 str 里面，返回 0。

mysql> select LOCATE('bar', 'foobarbar',5);

-> 7

这函数是多字节可靠的。

INSTR(str,substr)

返回子串 substr 在字符串 str 中的第一个出现的位置。这与有 2 个参数形式的 LOCATE()相同，除了参数被颠倒。

mysql> select INSTR('foobarbar', 'bar');

-> 4

mysql> select INSTR('xbar', 'foobar');

-> 0

这函数是多字节可靠的。

LPAD(str,len,padstr)

返回字符串 str，左面用字符串 padstr 填补直到 str 是 len 个字符长。

mysql> select LPAD('hi',4,'??');

-> '??hi'

RPAD(str,len,padstr)

返回字符串 str，右面用字符串 padstr 填补直到 str 是 len 个字符长。

mysql> select RPAD('hi',5,'?');

-> 'hi???'

LEFT(str,len)

返回字符串 str 的最左面 len 个字符。

mysql> select LEFT('foobarbar', 5);

-> 'fooba'

该函数是多字节可靠的。

RIGHT(str,len)

返回字符串 str 的最右面 len 个字符。

mysql> select RIGHT('foobarbar', 4);

-> 'rbar'

该函数是多字节可靠的。

SUBSTRING(str,pos,len)

SUBSTRING(str FROM pos FOR len)

MID(str,pos,len)

从字符串 str 返回一个 len 个字符的子串，从位置 pos 开始。使用 FROM 的变种形式是 ANSI SQL92 语法。

mysql> select SUBSTRING('Quadratically',5,6);

-> 'ratica'

该函数是多字节可靠的。

SUBSTRING(str,pos)

SUBSTRING(str FROM pos)

从字符串 str 的起始位置 pos 返回一个子串。

mysql> select SUBSTRING('Quadratically',5);

-> 'ratically'

mysql> select SUBSTRING('foobarbar' FROM 4);

-> 'barbar'

该函数是多字节可靠的。

SUBSTRING_INDEX(str,delim,count)

返回从字符串 str 的第 count 个出现的分隔符 delim 之后的子串。如果 count 是正数，返回最后的分隔符到左边(从左边数)的所有字符。如果 count 是负数，返回最后的分隔符到右边的所有字符(从右边数)。

mysql> select SUBSTRING_INDEX('www.mysql.com', '.', 2);

-> 'www.mysql'

mysql> select SUBSTRING_INDEX('www.mysql.com', '.', -2);

-> 'mysql.com'

该函数对多字节是可靠的。

LTRIM(str)

返回删除了其前置空格字符的字符串 str。

mysql> select LTRIM(' barbar');

-> 'barbar'

RTRIM(str)

返回删除了其拖后空格字符的字符串 str。

mysql> select RTRIM('barbar ');

-> 'barbar'

该函数对多字节是可靠的。

TRIM([[BOTH | LEA

DING | TRAILING] [remstr] FROM] str)

返回字符串 str，其所有 remstr 前缀或后缀被删除了。如果没有修饰符 BOTH、LEADING 或 TRAILING 给出，BOTH 被假定。如果 remstr 没被指定，空格被删除。

mysql> select TRIM(' bar ');

-> 'bar'

mysql> select TRIM(LEADING 'x' FROM 'xxxbarxxx');

-> 'barxxx'

mysql> select TRIM(BOTH 'x' FROM 'xxxbarxxx');

-> 'bar'

mysql> select TRIM(TRAILING 'xyz' FROM 'barxxyz');

-> 'barx'

该函数对多字节是可靠的。

SOUNDEX(str)

返回 str 的一个同音字符串。听起来“大致相同”的 2 个字符串应该有相同的同音字符串。一个“标准”的同音字符串长是 4 个字符，但是 SOUNDEX()函数返回一个任意长的字符串。你可以在结果上使用 SUBSTRING()得到一个“标准”的 同音串。所有非数字字母字符在给定的字符串中被忽略。所有在 A-Z 之外的字符国际字母被当作元音。

mysql> select SOUNDEX('Hello');

-> 'H400'

mysql> select SOUNDEX('Quadratically');

-> 'Q36324'

SPACE(N)

返回由 N 个空格字符组成的一个字符串。

mysql> select SPACE(6);

-> ' '

REPLACE(str,from_str,to_str)

返回字符串 str，其字符串 from_str 的所有出现由字符串 to_str 代替。

mysql> select REPLACE('www.mysql.com', 'w', 'Ww');

-> 'WwWwWw.mysql.com'

该函数对多字节是可靠的。

REPEAT(str,count)

返回由重复 countTimes 次的字符串 str 组成的一个字符串。如果 count <= 0，返回一个空字符串。如果 str 或 count 是 NULL，返回 NULL。

mysql> select REPEAT('MySQL', 3);

-> 'MySQLMySQLMySQL'

REVERSE(str)

返回颠倒字符顺序的字符串 str。

mysql> select REVERSE('abc');

-> 'cba'

该函数对多字节可靠的。

INSERT(str,pos,len,newstr)

返回字符串 str，在位置 pos 起始的子串且 len 个字符长得子串由字符串 newstr 代替。

mysql> select INSERT('Quadratic', 3, 4, 'What');

-> 'QuWhattic'

该函数对多字节是可靠的。

ELT(N,str1,str2,str3,...)

如果 N= 1，返回 str1，如果 N= 2，返回 str2，等等。如果 N 小于 1 或大于参数个数，返回 NULL。ELT()是 FIELD()反运算。

mysql> select ELT(1, 'ej', 'Heja', 'hej', 'foo');

-> 'ej'

mysql> select ELT(4, 'ej', 'Heja', 'hej', 'foo');

-> 'foo'

FIELD(str,str1,str2,str3,...)

返回 str 在 str1, str2, str3, ...清单的索引。如果 str 没找到，返回 0。FIELD()是 ELT()反运算。

mysql> select FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo');

-> 2

mysql> select FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo');

-> 0

FIND_IN_SET(str,strlist)

如果字符串 str 在由 N 子串组成的表 strlist 之中，返回一个 1 到 N 的值。一个字符串表是被“,”分隔的子串组成的一个字符串。如果第一个参数是一个常数字符串并且第二个参数是一种类型为 SET 的列，FIND_IN_SET()函数被优化而使用位运算！如果 str 不是在 strlist 里面或如果 strlist 是空字符串，返回 0。如果任何一个参数是 NULL，返回 NULL。如果第

一个参数包含一个“,”，该函数将工作不正常。

```
mysql> SELECT FIND_IN_SET('b','a,b,c,d');
    -> 2
```

MAKE_SET(bits,str1,str2,...)

返回一个集合 (包含由“,”字符分隔的子串组成的一个字符串)，由相应的位在 bits 集合中的的字符串组成。str1 对应于位 0，str2 对应位 1，等等。在 str1, str2, ...中的 NULL 串不添加到结果中。

```
mysql> SELECT MAKE_SET(1,'a','b','c');
    -> 'a'
mysql> SELECT MAKE_SET(1 | 4,'hello','nice','world');
    -> 'hello,world'
mysql> SELECT MAKE_SET(0,'a','b','c');
    -> ''
```

EXPORT_SET(bits,on,off,[separator,[number_of_bits]])

返回一个字符串，在这里对于在“bits”中设定每一位，你得到一个“on”字符串，并且对于每个复位(reset)的位，你得到一个“off”字符串。每个字符串用“separator”分隔(缺省“,”)，并且只有“bits”的“number_of_bits” (缺省 64)位被使用。

```
mysql> select EXPORT_SET(5,'Y','N',',',4)
    -> Y,N,Y,N
```

LCASE(str)

LOWER(str)

返回字符串 str，根据当前字符集映射(缺省是 ISO-8859-1 Latin1)把所有的字符改变成小写。该函数对多字节是可靠的。

```
mysql> select LCASE('QUADRATICALLY');
    -> 'quadratically'
```

UCASE(str)

UPPER(str)

返回字符串 str，根据当前字符集映射(缺省是 ISO-8859-1 Latin1)把所有的字符改变成大写。该函数对多字节是可靠的。

```
mysql> select UCASE('Hej');
    -> 'HEJ'
```

该函数对多字节是可靠的。

LOAD_FILE(file_name)

读入文件并且作为一个字符串返回文件内容。文件必须在服务器上，你必须指定到文件的完整路径名，而且你必须有 file 权限。文件必须所有内容都是可读的并且小于 max_allowed_packet。如果文件不存在或由于上面原因之一不能被读出，函数返回 NULL。

```
mysql> UPDATE table_name
      SET blob_column=LOAD_FILE("/tmp/picture")
      WHERE id=1;
```

MySQL 必要时自动变换数字为字符串，并且反过来也如此：

```
mysql> SELECT 1+"1";
```

-> 2

mysql> SELECT CONCAT(2,' test');

-> '2 test'

如果你想要明确地变换一个数字到一个字符串，把它作为参数传递到CONCAT()。

如果字符串函数提供一个二进制字符串作为参数，结果字符串也是一个二进制字符串。被变换到一个字符串的数字被当作是一个二进制字符串。这仅影响比较

mysql 时间函数用法集合

这里是一个使用日期函数的例子。下面的查询选择了所有记录，其 date_col 的值是在最后 30 天以内：

```
mysql> SELECT something FROM table
WHERE TO_DAYS(NOW()) - TO_DAYS(date_col) <= 30;
```

DAYOFWEEK(date)

返回日期 date 的星期索引(1=星期天，2=星期一，……7=星期六)。这些索引值对应于 ODBC 标准。

```
mysql> select DAYOFWEEK('1998-02-03');
-> 3
```

WEEKDAY(date)

返回 date 的星期索引(0=星期一，1=星期二，……6= 星期天)。

```
mysql> select WEEKDAY('1997-10-04 22:23:00');
-> 5
mysql> select WEEKDAY('1997-11-05');
-> 2
```

DAYOFMONTH(date)

返回 date 的月份中日期，在 1 到 31 范围内。

```
mysql> select DAYOFMONTH('1998-02-03');
-> 3
```

DAYOFYEAR(date)

返回 date 在一年中的日数，在 1 到 366 范围内。

```
mysql> select DAYOFYEAR('1998-02-03');
-> 34
```

MONTH(date)

返回 date 的月份，范围 1 到 12。

```
mysql> select MONTH('1998-02-03');
-> 2
```

DAYNAME(date)
返回 date 的星期名字。
mysql> select DAYNAME("1998-02-05");
-> 'Thursday'

MONTHNAME(date)
返回 date 的月份名字。
mysql> select MONTHNAME("1998-02-05");
-> 'February'

QUARTER(date)
返回 date 一年中的季度，范围 1 到 4。
mysql> select QUARTER('98-04-01');
-> 2

WEEK(date)

WEEK(date,first)
对于星期天是一周的第一天的地方，有一个单个参数，返回 date 的周数，范围在 0 到 52。2 个参数形式 WEEK()允许你指定星期是否开始于星期天或星期一。如果第二个参数是 0，星期从星期天开始，如果第二个参数是 1，从星期一开始。
mysql> select WEEK('1998-02-20');
-> 7
mysql> select WEEK('1998-02-20',0);
-> 7
mysql> select WEEK('1998-02-20',1);
-> 8

YEAR(date)
返回 date 的年份，范围在 1000 到 9999。
mysql> select YEAR('98-02-03');
-> 1998

HOUR(time)
返回 time 的小时，范围是 0 到 23。
mysql> select HOUR('10:05:03');
-> 10

MINUTE(time)
返回 time 的分钟，范围是 0 到 59。
mysql> select MINUTE('98-02-03 10:05:03');
-> 5

SECOND(time)
回来 time 的秒数，范围是 0 到 59。
mysql> select SECOND('10:05:03');
-> 3

PERIOD_ADD(P,N)
增加 N 个月到阶段 P（以格式 YYMM 或 YYYYMM)。以格式 YYYYMM 返回值。注意阶段参数 P 不是日期值。
mysql> select PERIOD_ADD(9801,2);
-> 199803

PERIOD_DIFF(P1,P2)
返回在时期 P1 和 P2 之间月数，P1 和 P2 应该以格式 YYMM 或 YYYYMM。注意，时期参数 P1 和 P2 不是日期值。
mysql> select PERIOD_DIFF(9802,199703);
-> 11

DATE_ADD(date,INTERVAL expr type)

DATE_SUB(date,INTERVAL expr type)

ADDDATE(date,INTERVAL expr type)

SUBDATE(date,INTERVAL expr type)
这些功能执行日期运算。对于 MySQL 3.22，他们是新的。ADDDATE()和 SUBDATE()是 DATE_ADD()和 DATE_SUB()的同义词。
在 MySQL 3.23 中，你可以使用+和-而不是 DATE_ADD()和 DATE_SUB()。（见例子）date 是一个指定开始日期的
DATETIME 或 DATE 值，expr 是指定加到开始日期或从开始日期减去的间隔值一个表达式，expr 是一个字符串；它可以以
一个“-”开始表示负间隔。type 是一个关键词，指明表达式应该如何被解释。EXTRACT(type FROM date)函数从日期
中返回“type”间隔。下表显示了 type 和 expr 参数怎样被关联： type 值 含义 期望的 expr 格式
SECOND 秒 SECONDS
MINUTE 分钟 MINUTES
HOUR 时间 HOURS
DAY 天 DAYS
MONTH 月 MONTHS
YEAR 年 YEARS
MINUTE_SECOND 分钟和秒 "MINUTES:SECONDS"
HOUR_MINUTE 小时和分钟 "HOURS:MINUTES"
DAY_HOUR 天和小时 "DAYS HOURS"
YEAR_MONTH 年和月 "YEARS-MONTHS"
HOUR_SECOND 小时, 分钟, "HOURS:MINUTES:SECONDS"

DAY_MINUTE 天, 小时, 分钟 "DAYS HOURS:MINUTES"
DAY_SECOND 天, 小时, 分钟, 秒 "DAYS HOURS:MINUTES:SECONDS"

MySQL 在 expr 格式中允许任何标点分隔符。表示显示的是建议的分隔符。如果 date 参数是一个 DATE 值并且你的计算仅仅包含 YEAR、MONTH 和 DAY 部分(即,没有时间部分),结果是一个 DATE 值。否则结果是一个 DATETIME 值。

```
mysql> SELECT "1997-12-31 23:59:59" + INTERVAL 1 SECOND;
-> 1998-01-01 00:00:00
mysql> SELECT INTERVAL 1 DAY + "1997-12-31";
-> 1998-01-01
mysql> SELECT "1998-01-01" - INTERVAL 1 SECOND;
-> 1997-12-31 23:59:59
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
INTERVAL 1 SECOND);
-> 1998-01-01 00:00:00
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
INTERVAL 1 DAY);
-> 1998-01-01 23:59:59
mysql> SELECT DATE_ADD("1997-12-31 23:59:59",
INTERVAL "1:1" MINUTE_SECOND);
-> 1998-01-01 00:01:00
mysql> SELECT DATE_SUB("1998-01-01 00:00:00",
INTERVAL "1 1:1:1" DAY_SECOND);
-> 1997-12-30 22:58:59
mysql> SELECT DATE_ADD("1998-01-01 00:00:00",
INTERVAL "-1 10" DAY_HOUR);
-> 1997-12-30 14:00:00
mysql> SELECT DATE_SUB("1998-01-02", INTERVAL 31 DAY);
-> 1997-12-02
mysql> SELECT EXTRACT(YEAR FROM "1999-07-02");
-> 1999
mysql> SELECT EXTRACT(YEAR_MONTH FROM "1999-07-02 01:02:03");
-> 199907
mysql> SELECT EXTRACT(DAY_MINUTE FROM "1999-07-02 01:02:03");
-> 20102
```

如果你指定太短的间隔值(不包括 type 关键词期望的间隔部分),MySQL 假设你省掉了间隔值的最左面部分。例如,如果你指定一个 type 是 DAY_SECOND,值 expr 被希望有天、小时、分钟和秒部分。如果你象"1:10"这样指定值,MySQL 假设日子和小时部分是丢失的并且值代表分钟和秒。换句话说,"1:10" DAY_SECOND 以它等价于"1:10" MINUTE_SECOND 的方式解释,这对那 MySQL 解释 TIME 值表示经过的时间而非作为一天的时间的方式有二义性。如果你使用确实不正确的日期,结果是 NULL。如果你增加 MONTH、YEAR_MONTH 或 YEAR 并且结果日期大于新月份的最大值天数,日子在新月用最大的天调整。

```
mysql> select DATE_ADD('1998-01-30', Interval 1 month);
-> 1998-02-28
```

注意，从前面的例子中词 INTERVAL 和 type 关键词不是区分大小写的。

TO_DAYS(date)

给出一个日期 date，返回一个天数(从 0 年的天数)。

```
mysql> select TO_DAYS(950501);
-> 728779
mysql> select TO_DAYS('1997-10-07');
-> 729669
```

TO_DAYS()

不打算用于使用格列高里历(1582)出现前的值。

FROM_DAYS(N)

给出一个天数 N，返回一个 DATE 值。

```
mysql> select FROM_DAYS(729669);
-> '1997-10-07'
```

DATE_FORMAT(date,format)

根据 format 字符串格式化 date 值。下列修饰符可以被用在 format 字符串中：

%M 月名字(January……December)
%W 星期名字(Sunday……Saturday)
%D 有英语前缀的月份的日期(1st, 2nd, 3rd, 等等。)
%Y 年, 数字, 4 位
%y 年, 数字, 2 位
%a 缩写的星期名字(Sun……Sat)
%d 月份中的天数, 数字(00……31)
%e 月份中的天数, 数字(0……31)
%m 月, 数字(01……12)
%c 月, 数字(1……12)
%b 缩写的月份名字(Jan……Dec)
%j 一年中的天数(001……366)
%H 小时(00……23)
%k 小时(0……23)
%h 小时(01……12)
%I 小时(01……12)
%l 小时(1……12)
%i 分钟, 数字(00……59)
%r 时间,12 小时(hh:mm:ss [AP]M)
%T 时间,24 小时(hh:mm:ss)
%S 秒(00……59)
%s 秒(00……59)
%p AM 或 PM

%w 一个星期中的天数(0=Sunday ……6=Saturday)
%U 星期(0……52), 这里星期天是星期的第一天
%u 星期(0……52), 这里星期一是星期的第一天
%% 一个文字“%”。

所有的其他字符不做解释被复制到结果中。

```
mysql> select DATE_FORMAT('1997-10-04 22:23:00', '%W %M %Y');
-> 'Saturday October 1997'
mysql> select DATE_FORMAT('1997-10-04 22:23:00', '%H:%i:%s');
-> '22:23:00'
mysql> select DATE_FORMAT('1997-10-04 22:23:00',
'%D %y %a %d %m %b %j');
-> '4th 97 Sat 04 10 Oct 277'
mysql> select DATE_FORMAT('1997-10-04 22:23:00',
'%H %k %I %r %T %S %w');
-> '22 22 10 10:23:00 PM 22:23:00 00 6'
```

MySQL3.23 中，在格式修饰符字符前需要%。在 MySQL 更早的版本中，%是可选的。

TIME_FORMAT(time,format)

这象上面的 DATE_FORMAT()函数一样使用，但是 format 字符串只能包含处理小时、分钟和秒的那些格式修饰符。其他修饰符产生一个 NULL 值或 0。

CURDATE()

CURRENT_DATE

以'YYYY-MM-DD'或 YYYYMMDD 格式返回今天日期值，取决于函数是在一个字符串还是数字上下文被使用。

```
mysql> select CURDATE();
-> '1997-12-15'
mysql> select CURDATE() + 0;
-> 19971215
```

CURTIME()

CURRENT_TIME

以'HH:MM:SS'或 HHMMSS 格式返回当前时间值，取决于函数是在一个字符串还是在数字的上下文被使用。

```
mysql> select CURTIME();
-> '23:50:26'
mysql> select CURTIME() + 0;
-> 235026
```

NOW()

SYSDATE()

CURRENT_TIMESTAMP

以'YYYY-MM-DD HH:MM:SS'或 YYYYMMDDHHMMSS 格式返回当前的日期和时间，取决于函数是在一个字符串还是在数字的上下文被使用。

```
mysql> select NOW();
-> '1997-12-15 23:50:26'
mysql> select NOW() + 0;
-> 19971215235026
```

UNIX_TIMESTAMP()

UNIX_TIMESTAMP(date)

如果没有参数调用，返回一个 Unix 时间戳记(从'1970-01-01 00:00:00'GMT 开始的秒数)。如果 UNIX_TIMESTAMP()用一个 date 参数被调用，它返回从'1970-01-01 00:00:00' GMT 开始的秒数值。date 可以是一个 DATE 字符串、一个 DATETIME 字符串、一个 TIMESTAMP 或以 YYMMDD 或 YYYYMMDD 格式的本地时间的一个数字。

```
mysql> select UNIX_TIMESTAMP();
-> 882226357
mysql> select UNIX_TIMESTAMP('1997-10-04 22:23:00');
-> 875996580
```

当 UNIX_TIMESTAMP 被用于一个 TIMESTAMP 列，函数将直接接受值，没有隐含的“string-to-unix-timestamp”变换。

FROM_UNIXTIME(unix_timestamp)

以'YYYY-MM-DD HH:MM:SS'或 YYYYMMDDHHMMSS 格式返回 unix_timestamp 参数所表示的值，取决于函数是在一个字符串还是或数字上下文中被使用。

```
mysql> select FROM_UNIXTIME(875996580);
-> '1997-10-04 22:23:00'
mysql> select FROM_UNIXTIME(875996580) + 0;
-> 19971004222300
```

FROM_UNIXTIME(unix_timestamp,format)

返回表示 Unix 时间标记的一个字符串，根据 format 字符串格式化。format 可以包含与 DATE_FORMAT()函数列出的条目同样的修饰符。

```
mysql> select FROM_UNIXTIME(UNIX_TIMESTAMP(),
'%Y %D %M %h:%i:%s %x');
-> '1997 23rd December 03:43:30 x'
```

SEC_TO_TIME(seconds)

返回 seconds 参数，变换成小时、分钟和秒，值以'HH:MM:SS'或 HHMMSS 格式化，取决于函数是在一个字符串还是在数字上下文中被使用。

```
mysql> select SEC_TO_TIME(2378);
-> '00:39:38'
mysql> select SEC_TO_TIME(2378) + 0;
-> 3938
```

TIME_TO_SEC(time)

返回 time 参数，转换成秒。

```
mysql> select TIME_TO_SEC('22:23:00');
-> 80580
mysql> select TIME_TO_SEC('00:39:38');
-> 2378
```

Mysql 取系统函数：

Select curtime();

Select curdate():

Select sysdate():

select now();

一、 控制流程函数

a) CASE WHEN THEN 函数

语法： CASE value WHEN [compare-value] THEN result [WHEN [compare-value] THEN result ……] [ELSE result] END CASE WHEN [condition] THEN result [WHEN[condition] THEN result ……] [ELSE result] END ；

函数用法说明：在第一个方案的返回结果中， value =compare-value 。而第二个方案的返回结果是第一种情况的真实结果。如果没有匹配的结果值，则返回结果为 ELSE 后的结果，如果没有 ELSE 部分，则返回值为 NULL

b) IF 函数用法

语法： IF(expr1,expr2,expr3)

函数用法说明：如果 expr1 是 TRUE (expr1 <> 0 and expr1 <> NULL) ，则 IF() 的返回值为 expr2 ; 否则返回值则为 expr3 。 IF() 的返回值为数字值或字符串值，具体情况视其所在语境而定

c) IFNULL 函数

语法： IFNULL(expr1,expr2)

函数用法说明：假如 expr1 不为 NULL ，则 IFNULL() 的返回值为 expr1 ; 否则其返回值为 expr2 。 IFNULL() 的返回值是数字或是字符串，具体情况取决于其所使用的语境

二、 字符串比较函数

a) 函数 ascii(str)

函数用法说明：返回值为字符串 str 的最左字符的数值。假如 str 为空字符串，则返回值为 0 。假如 str 为 NULL ，则返回值为 NULL 。 ASCII() 用于带有从 0 到 255 的数值的字符

b) 函数 BIN(N)

函数用法说明：返回值为 N 的二进制值的字符串表示，其中 N 为一个 longlong (BIGINT) 数字。这等同于 CONV(N ,10,2) 。假如 N 为 NULL ，则返回值为 NULL 。

c) 函数 CHAR(N ,... [USING charset])

函数用法说明： CHAR() 将每个参数 N 理解为一个整数，其返回值为一个包含这些整数的代码值所给出的字符的字符串。 NULL 值被省略。

d) 函数 CHAR_LENGTH(str)

函数使用说明：返回值为字符串 str 的长度，长度的单位为字符。一个多字节字符算作一个单字符。对于一个 包含五个二字节字符集 , LENGTH() 返回值为 10, 而 CHAR_LENGTH() 的返回值为 5

e) 函数 CHARACTER_LENGTH(str)

函数使用说明： CHARACTER_LENGTH() 是 CHAR_LENGTH() 的同义词。

f) 函数 COMPRESS(string_to_compress)

函数使用说明： COMPRESS(压缩一个字符串。这个函数要求 MySQL 已经用一个

诸如 zlib 的压缩库压缩过。 否则，返回值始终是 NULL 。 UNCOMPRESS() 可将压缩过的字符串进行解压缩) 。

g) 函数 CONCAT(str1 ,str2 ,...)

函数使用说明：返回结果为连接参数产生的字符串。如有任何一个参数为 NULL ，则

返回值为 NULL 。或许有一个或多个参数。 如果所有参数均为非二进制字符串，则结果为非二进制字符串。 如果自变量中含有任一二进制字符串，则结果为一个二进制字符串。一个数字参数被转化为与之相等的二进制字符串格式；若要避免这种情况，可使用显式类型 cast, 例如： SELECT CONCAT(CAST(int_col AS CHAR), char_col)

h) 函数 CONCAT_WS(separator ,str1 ,str2 ,...)

函数使用说明： CONCAT_WS() 代表 CONCAT With Separator ，是 CONCAT() 的

特殊形式。 第一个参数是其它参数的分隔符。分隔符的位置放在要连接的两个字符串之间。分隔符可以是一个字符串，也可以是其它参数。如果分隔符为 NULL ，则结果为 NULL 。函数会忽略任何分隔符参数后的 NULL 值。

i) 函数 CONV(N from_base, to_base)

函数使用说明：不同数基间转换数字。返回值为数字的 N 字符串表示，由 from_base 基转化为 to_base 基。如有任意一个参数为 NULL ，则返回值为 NULL 。自变量 N 被理解为

一个整数，但是可以被指定为一个整数或字符串。最小基数为 2 ，而最大基数则为 36 。 If to_base 是一个负数，则 N 被看作一个带符号数。否则， N 被看作无符号数。 CONV() 的运行精确度为 64 比特。

j) 函数 ELT(N ,str1 ,str2 ,str3 ,...)

函数使用说明：若 N = 1 ，则返回值为 str1 ，若 N = 2 ，则返回值为 str2 ，以此

类推。 若 N 小于 1 或大于参数的数目，则返回值为 NULL 。ELT() 是 FIELD() 的补数

k) 函数 EXPORT_SET(bits ,on ,off [,separator [,number_of_bits]])

函数使用说明： 返回值为一个字符串，其中对于 bits 值中的每个位组，可以得到

一个 on 字符串，而对于每个清零比特位，可以得到一个 off 字符串。 bits 中的比特值按照从右到左的顺序接受检验 (由低位比特到高位比特) 。字符串被分隔字符串分开 (默认为逗号 ‘,’) ，按照从左到右的顺序被添加到结果中。 number_of_bits 会给出被检验的二进制位数 (默认为 64) 。

l) 函数 FIELD(str, str1, str2, str3, …...)

函数使用说明：返回值为 str1 , str2 , str3 ,…… 列表中的 str 指数。在找不到 str 的情况下，返回值为 0 。如果所有对于 FIELD() 的参数均为字符串，则所有参数均按照字符串进行比较。如果所有的参数均为数字，则按照数字进行比较。否则，参数按照双倍进行比较。如果 str 为 NULL ，则返回值为 0 ，原因是 NULL 不能同任何值进行同等比较。 FIELD() 是 ELT() 的补数。

m) 函数 FIND_IN_SET(str, strlist)

函数使用说明： 假如字符串 str 在由 N 子链组成的字符串列表 strlist 中， 则返

回值的范围在 1 到 N 之间 。一个字符串列表就是一个由一些被 ‘,’ 符号分开的自链组成的字符串。如果第一个参数是一个常数字符串，而第二个是 type SET 列，则 FIND_IN_SET() 函数被优化，使用比特计算。如果 str 不在 strlist 或 strlist 为空字符串，则返回值为 0 。如任意一个参数为 NULL ，则返回值为 NULL 。 这个函数在第一个参数包含一个逗号 (‘,’) 时将无法正常运行。

n) 函数 FORMAT(X ,D)

函数使用说明： 将 number X 设置为格式 '#,###,###.##', 以四舍五入的方式保留到小数点后 D 位 , 而返回结果为一个字符串。

o) 函数 HEX(N_or_S)

函数使用说明：如果 N_OR_S 是一个数字，则返回一个 十六进制值 N 的 字符串表示，在这里， N 是一个 longlong (BIGINT) 数。这相当于 CONV(N,10,16) 。如果 N_OR_S 是一个字符串，则返回值为一个 N_OR_S 的十六进制字符串表示，其中每个 N_OR_S 里的每个字符被转化为两个十六进制数字。

p) 函数 INSTR(str,substr)

函数使用说明：返回字符串 str 中子字符串的第一个出现位置。这和 LOCATE() 的双参数形式相同，除非参数的顺序被颠倒。

q) 函数 LCASE(str)

函数使用说明：LCASE() 是 LOWER() 的同义词

r) 函数 LEFT(str,len)

函数使用说明：返回从字符串 str 开始的 len 最左字符

s) 函数 LENGTH(str)

函数使用说明： 返回值为字符串 str 的长度，单位为字节。一个多字节字符算作多字节。这意味着 对于一个包含 5 个 2 字节字符的字符串， LENGTH() 的返回值为 10, 而 CHAR_LENGTH() 的返回值则为 5 。

t) 函数 LOAD_FILE(file_name)

函数使用说明：读取文件并将这一文件按照字符串的格式返回。 文件的位置必须在服务器上 , 你必须为文件制定路径全名，而且你还必须拥有 FILE 特许权。文件必须可读取，文件容量必须小于 max_allowed_packet 字节。若文件不存在，或因不满足上述条件而不能被读取， 则函数返回值为 NULL

u) 函数 LOCATE(substr ,str) , LOCATE(substr ,str ,pos)

函数使用说明：第一个语法返回字符串 str 中子字符串 substr 的第一个出现位置。第二个语法返回字符串 str 中子字符串 substr 的第一个出现位置, 起始位置在 pos 。如若 substr 不在 str 中，则返回值为 0 。

v) 函数 LOWER(str)

函数使用说明：返回字符串 str 以及所有根据最新的字符集映射表变为小写字母的字符

w) 函数 LPAD(str ,len ,padstr)

函数使用说明：返回字符串 str , 其左边由字符串 padstr 填补到 len 字符长度。假如 str 的长度大于 len , 则返回值被缩短至 len 字符。

x) 函数 LTRIM(str)

函数使用说明：返回字符串 str ，其引导空格字符被删除。

y) 函数 MAKE_SET(bits ,str1 ,str2 ,...)

函数使用说明： 返回一个设定值 (一个包含被 ‘,’ 号分开的字字符串的字符串) ，由在 bits 组中具有相应的比特的字符串组成。 str1 对应比特 0, str2 对应比特 1, 以此类推。str1 , str2 , ... 中的 NULL 值不会被添加到结果中。

z) 函数 MID(str ,pos ,len)

函数使用说明： MID(str ,pos ,len) 是 SUBSTRING(str ,pos ,len) 的同义词。

aa) 函数 OCT(N)
函数使用说明：返回一个 N 的八进制值的字符串表示，其中 N 是一个 longlong (BIGINT) 数。这等同于 CONV(N,10,8) 。若 N 为 NULL ，则返回值为 NULL 。

bb) 函数 OCTET_LENGTH(str)

函数使用说明： OCTET_LENGTH() 是 LENGTH() 的同义词。

cc) 函数 ORD(str)

函数使用说明：若字符串 str 的最左字符是一个多字节字符，则返回该字符的代码， 代码的计算通过使用以下公式计算其组成字节的数值而得出:

(1st byte code)

+ (2nd byte code × 256)

+ (3rd byte code × 2562

) ...

假如最左字符不是一个多字节字符，那么 ORD() 和函数 ASCII() 返回相同的值

dd) 函数 POSITION(substr IN str)

函数使用说明：POSITION(substr IN str) 是 LOCATE(substr ,str) 同义词

ee) 函数 QUOTE(str)

函数使用说明：引证一个字符串，由此产生一个在 SQL 语句中可用作完全转义数据值的结果。 返回的字符串由单引号标注，每例都带有单引号（‘’）、反斜线符号（‘\’）、ASCII NUL 以及前面有反斜线符号的 Control-Z 。如果自变量的值为 NULL, 则返回不带单引号的单词 “NULL” 。

ff) 函数 REPEAT(str ,count)

函数使用说明：返回一个由重复的字符串 str 组成的字符串，字符串 str 的数目等于 count 。若 count <= 0, 则返回一个空字符串。若 str 或 count 为 NULL ，则返回 NULL 。

gg) 函数 REPLACE(str ,from_str ,to_str)

函数使用说明：返回字符串 str 以及所有被字符串 to_str 替代的字符串 from_str 。

hh) 函数 REVERSE(str)

函数使用说明：返回字符串 str ，顺序和字符顺序相反。

ii) 函数 RIGHT(str ,len)

函数使用说明：从字符串 str 开始，返回最右 len 字符。

jj) 函数 RPAD(str ,len ,padstr)

函数使用说明：返回字符串 str , 其右边被字符串 padstr 填补至 len 字符长度。假如字符串 str 的长度大于 len , 则返回值被缩短到与 len 字符相同长度

kk) 函数 RTRIM(str)

函数使用说明：返回字符串 str ，结尾空格字符被删去。

ll) 函数 SOUNDEX(str)

函数使用说明：从 str 返回一个 soundex 字符串。 两个具有几乎同样探测的字符串应该具有同样的 soundex 字符串。一个标准的 soundex 字符串的长度为 4 个字符，然而 SOUNDEX() 函数会返回一个人以长度的字符串。 可使用结果中的 SUBSTRING() 来得到

一个标准 soundex 字符串。在 str 中， 会忽略所有未按照字母顺序排列的字符。所有不在 A-Z 范围之内的国际字母符号被视为元音字母。

mm) 函数 expr1 SOUNDS LIKE expr2

函数使用说明： 这相当于 SOUNDEX(expr1) = SOUNDEX(expr2) 。

nn) 函数 SPACE(N)

函数使用说明：返回一个由 N 间隔符号组成的字符串

oo) 函数 SUBSTRING(str ,pos) , SUBSTRING(str FROM pos) SUBSTRING(str ,pos ,len) , SUBSTRING(str FROM pos FOR len)

函数使用说明：不带有 len 参数的格式从字符串 str 返回一个子字符串，起始于位置 pos 。带有 len 参数的格式从字符串 str 返回一个长度同 len 字符相同的子字符串，起始于位置 pos 。 使用 FROM 的格式为标准 SQL 语法。也可能对 pos 使用一个负值。假若这样，则子字符串的位置起始于字符串结尾的 pos 字符，而不是字符串的开头位置。在以下格式的函数中可以对 pos 使用一个负值。

pp) 函数 SUBSTRING_INDEX(str ,delim ,count)

函数使用说明:在定界符 delim 以及 count 出现前,从字符串 str 返回自字符串。若 count 为正值, 则返回最终定界符(从左边开始) 左边的一切内容。若 count 为负值，则返回定界符（从右边开始）右边的一切内容。

qq) 函数 TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str) TRIM(remstr FROM] str)

函数使用说明：返回字符串 str ， 其中所有 remstr 前缀和/ 或后缀都已被删除。若分类符 BOTH 、LEADIN 或 TRAILING 中没有一个是给定的, 则假设为 BOTH 。 remstr 为可选项，在未指定情况下，可删除空格

rr) 函数 UCASE(str)

函数使用说明：UCASE() 是 UPPER() 的同义词

ss) 函数 UNCOMPRESS(string_to_uncompress)

函数使用说明：对经 COMPRESS() 函数压缩后的字符串进行解压缩。若参数为压缩值，则结果为 NULL 。这个函数要求 MySQL 已被诸如 zlib 之类的压缩库编译过。否则, 返回值将始终是 NULL

tt) 函数 UNCOMPRESSED_LENGTH(compressed_string)

函数使用说明： 返回压缩字符串压缩前的长度。

uu) 函数 UNHEX(str)

函数使用说明：执行从 HEX(str) 的反向操作。就是说，它将参数中的每一对十六进制数字理解为一个数字，并将其转化为该数字代表的字符。结果字符以二进制字符串的形式返回

vv) 函数 UPPER(str)

函数使用说明：返回字符串 str ， 以及根据最新字符集映射转化为大写字母的字符

三、 数学函数

a) 函数 ABS(X)

函数使用说明：返回 X 的绝对值

b) 函数 ACOS(X)

函数使用说明：返回 X 反余弦 , 即 , 余弦是 X 的值。若 X 不在 -1 到 1 的范围之内，则返回 NULL 。

c) 函数 ASIN （ X ）

函数使用说明：返回 X 的反正弦，即，正弦为 X 的值。若 X 若 X 不在-1 到 1 的范围之内，则返回 NULL 。

d) 函数 ATAN(X)

函数使用说明：返回 X 的反正切，即，正切为 X 的值。

e) 函数 ATAN(Y ,X) , ATAN2(Y ,X)

函数使用说明：返回两个变量 X 及 Y 的反正切。 它类似于 Y 或 X 的反正切计

算 ， 除非两个参数的符号均用于确定结果所在象限。

f) 函数 CEILING(X) CEIL(X)

函数使用说明：返回不小于 X 的最小整数值。

g) 函数 COS(X)

函数使用说明：返回 X 的余弦，其中 X 在弧度上已知。

h) 函数 COT(X)

函数使用说明：返回 X 的余切

i) 函数 CRC32(expr)

函数使用说明：计算循环冗余码校验值并返回一个 32 比特无符号值。若参数为 NULL ，则结果为 NULL 。该参数应为一个字符串，而且在不是字符串的情况下会被作为字符串处理（若有可能）

j) 函数 DEGREES(X)

函数使用说明：返回参数 X，该参数由弧度被转化为度。

k) 函数 EXP(X)

函数使用说明：返回 e 的 X 乘方后的值 (自然对数的底) 。

l) 函数 FLOOR(X)

函数使用说明：返回不大于 X 的最大整数值 。

m) 函数 FORMAT(X ,D)

函数使用说明:将数字 X 的格式写成 '#,###,###.##' 格式 ，即保留小数点后 D 位,而第 D 位的保留方式为四舍五入，然后将结果以字符串的形式返回

n) 函数 LN(X)

函数使用说明：返回 X 的自然对数 ，即 ,X 相对于基数 e 的对数

o) 函数 LOG(X) LOG(B ,X)

函数使用说明：若用一个参数调用，这个函数就会返回 X 的自然对数。

p) 函数 LOG2(X)

函数使用说明：返回 X 的基数为 2 的对数。

q) 函数 LOG10(X)

函数使用说明：返回 X 的基数为 10 的对数。

r) 函数 MOD(N ,M) , N % M N MOD M

函数使用说明： 模操作。返回 N 被 M 除后的余数。

s) 函数 PI()

函数使用说明：返回 ω (pi) 的值。默认的显示小数位数是 7 位 , 然而 MySQL 内部会使用完全双精度值。

t) 函数 POW(X ,Y) , POWER(X ,Y)

函数使用说明：返回 X 的 Y 乘方的结果值。

u) 函数 RADIANS(X)

函数使用说明：返回由度转化为弧度的参数 X， (注意 ω 弧度等于 180 度)。

v) 函数 RAND() RAND(N)

函数使用说明：返回一个随机浮点值 v ，范围在 0 到 1 之间 (即 , 其范围为 0 ≤ v ≤ 1.0) 。若已指定一个整数参数 N ，则它被用作种子值，用来产生重复序列。

w) 函数 ROUND(X) ROUND(X ,D)

函数使用说明：返回参数 X, 其值接近于最近似的整数。在有两个参数的情况下，返回 X ，其值保留到小数点后 D 位，而第 D 位的保留方式为四舍五入。若要接保留 X 值小数点左边的 D 位，可将 D 设为负值。

x) 函数 SIGN(X)

函数使用说明：返回参数作为 -1 、 0 或 1 的符号，该符号取决于 X 的值为负、零或正。

y) 函数 SIN(X)

函数使用说明：返回 X 正弦，其中 X 在弧度中被给定。

z) 函数 SQRT(X)

函数使用说明： 返回非负数 X 的二次方根。

aa) 函数 TAN(X)

函数使用说明： 返回 X 的正切，其中 X 在弧度中被给定。

bb) 函数 TRUNCATE(X ,D)

函数使用说明： 返回被舍去至小数点后 D 位的数字 X 。若 D 的值为 0, 则结果

不带有小数点或不带有小数部分。可以将 D 设为负数 , 若要截去 (归零) X 小数点左起第 D 位开始后面所有低位的值

四、 日期和时间函数

a) 函数 ADDDATE(date ,INTERVAL expr type) ADDDATE(expr ,days)

函数使用说明： 当被第二个参数的 INTERVAL 格式激活后， ADDDATE() 就是 DATE_ADD() 的同义词。相关函数 SUBDATE() 则是 DATE_SUB() 的同义词。对于 INTERVAL 参数上的信息 ，请参见关于 DATE_ADD() 的论述。

b) 函数 ADDTIME(expr ,expr2)

函数使用说明：ADDTIME() 将 expr2 添加至 expr 然后返回结果。 expr 是一个时间或时间日期表达式，而 expr2 是一个时间表达式。

c) 函数 CONVERT_TZ(dt ,from_tz ,to_tz)

函数使用说明：CONVERT_TZ() 将时间日期值 dt 从 from_tz 给出的时区转到 to_tz 给出的时区，然后返回结果值。关于可能指定的时区的详细论述，若自变量无效，则这个函数会返回 NULL

d) 函数 CURDATE()

函数使用说明：将当前日期按照 'YYYY-MM-DD' 或 YYYYMMDD 格式的值返回，具体格式根据函数用在字符串或是数字语境中而定。

e) 函数 CURRENT_DATE CURRENT_DATE()

函数使用说明： CURRENT_DATE 和 CURRENT_DATE() 是的同义词 .

f) 函数 CURTIME()

函数使用说明： 将当前时间以 'HH:MM:SS' 或 HHMMSS 的格式返回， 具体格式根据函数用在字符串或是数字语境中而定。

g) 函数 CURRENT_TIME, CURRENT_TIME()

函数使用说明： CURRENT_TIME 和 CURRENT_TIME() 是 CURTIME() 的同义词。

h) 函数 CURRENT_TIMESTAMP, CURRENT_TIMESTAMP()

函数使用说明： CURRENT_TIMESTAMP 和 CURRENT_TIMESTAMP() 是 NOW() 的同义词

i) 函数 DATE(expr)

函数使用说明： 提取日期或时间日期表达式 expr 中的日期部分。

j) 函数 DATEDIFF(expr ,expr2)

函数使用说明：DATEDIFF() 返回起始时间 expr 和结束时间 expr2 之间的天数。Expr 和 expr2 为日期或 date-and-time 表达式。计算中只用到这些值的日期部分。

k) 函数 DATE_ADD(date ,INTERVAL expr type) DATE_SUB(date ,INTERVAL expr type)

函数使用说明：这些函数执行日期运算。 date 是一个 DATETIME 或 DATE 值，用来指定起始时间。 expr 是一个表达式，用来指定从起始日期添加或减去的时间间隔值。 Expr 是一个字符串 ; 对于负值的时间间隔，它可以以一个 ‘-’ 开头。 type 为关键词，它指示了表达式被解释的方式。

l) 函数 DATE_FORMAT(date ,format)

函数使用说明：根据 format 字符串安排 date 值的格式。

m) 函数 DAY(date)

函数使用说明： DAY() 和 DAYOFMONTH() 的意义相同

n) 函数 DAYNAME(date)

函数使用说明：返回 date 对应的工作日名称。

o) 函数 DAYOFMONTH(date)

函数使用说明：返回 date 对应的该月日期，范围是从 1 到 31

p) 函数 DAYOFWEEK(date)

函数使用说明：返回 date (1 = 周日 , 2 = 周一 , ..., 7 = 周六) 对应的工作日索引。这些索引值符合 ODBC 标准

q) 函数 DAYOFYEAR(date)

函数使用说明：返回 date 对应的一年中的天数，范围是从 1 到 366 。

r) 函数 EXTRACT(type FROM date)

函数使用说明： EXTRACT() 函数所使用的时间间隔类型说明符同 DATE_ADD() 或 DATE_SUB() 的相同 , 但它从日期中提取其部分，而不是执行日期运算。

s) 函数 FROM_DAYS(N)

函数使用说明： 给定一个天数 N , 返回一个 DATE 值。

t) 函数 FROM_UNIXTIME(unix_timestamp)ROM_UNIXTIME(unix_timestamp ,format)

函数使用说明：返回'YYYY-MM-DD HH:MM:SS' 或 YYYYMMDDHHMMSS 格式值的 unix_timestamp 参数表示，具体格式取决于该函数是否用在字符串中或是数字语境中。 若 format 已经给出，则结果的格式是根据 format 字符串而定。 format 可以包含同 DATE_FORMAT() 函数输入项列表中相同的说明符。

u) 函数 GET_FORMAT(DATE|TIME|DATETIME, 'EUR'|'USA'|'JIS'|'ISO'|'INTERNAL')

函数使用说明:返回一个格式字符串。这个函数在同 DATE_FORMAT() 及 STR_TO_DATE() 函数结合时很有用

v) 函数 HOUR(time)

函数使用说明：返回 time 对应的小时数。对于日时值的返回值范围是从 0 到 23

w) 函数 LAST_DAY(date)

函数使用说明：获取一个日期或日期时间值，返回该月最后一天对应的值。若参数无效,则返回 NULL 。

x) 函数 LOCALTIME, LOCALTIME()

函数使用说明： LOCALTIME 及 LOCALTIME() 和 NOW() 具有相同意义。

y) 函数 LOCALTIMESTAMP, LOCALTIMESTAMP()

函数使用说明： LOCALTIMESTAMP 和 LOCALTIMESTAMP() 和 NOW() 具有相同意义。

z) 函数 MAKEDATE(year ,dayofyear)

函数使用说明：给出年份值和一年中的天数值，返回一个日期。 dayofyear 必须大于 0 ，否则结果为 NULL 。

aa) 函数 MAKETIME(hour ,minute ,second)

函数使用说明： 返回由 hour 、 minute 和 second 参数计算得出的时间值

bb) 函数 CROSECOND(expr)

函数使用说明：从时间或日期时间表达式 expr 返回微秒值，其数字范围从 0 到 999999 。

cc) 函数 MINUTE(time)

函数使用说明：返回 time 对应的分钟数 , 范围是从 0 到 59 。

dd) 函数 MONTH(date)

函数使用说明：返回 date 对应的月份，范围时从 1 到 12 。

ee) 函数 MONTHNAME(date)

函数使用说明： 返回 date 对应月份的全名

ff) 函数 NOW()

函数使用说明：返回当前日期和时间值，其格式为 'YYYY-MM-DD HH:MM:SS' 或 YYYYMMDDHHMMSS ， 具体格式取决于该函数是否用在字符串中或数字语境中。

gg) 函数 PERIOD_ADD(P ,N)

函数使用说明：添加 N 个月至周期 P (格式为 YYMM 或 YYYYMM) ，返回值的格式为 YYYYMM 。注意周期参数 P 不是 日期值。

hh) 函数 PERIOD_DIFF(P1 ,P2)

函数使用说明：返回周期 P1 和 P2 之间的月份数。 P1 和 P2 的格式应该为 YYMM 或 YYYYMM 。注意周期参数 P1 和 P2 不是 日期值。

ii) 函数 QUARTER(date)

函数使用说明：返回 date 对应的一年中的季度值，范围是从 1 到 4

jj) 函数 SECOND(time)

函数使用说明：返回 time 对应的秒数 , 范围是从 0 到 59 。

kk) 函数 SEC_TO_TIME(seconds)

函数使用说明： 返回被转化为小时、 分钟和秒数的 seconds 参数值 , 其格式为 'HH:MM:SS' 或 HHMMSS ，具体格式根据该函数是否用在字符串或数字语境中而定

ll) 函数 STR_TO_DATE(str ,format)

函数使用说明：这是 DATE_FORMAT() 函数的倒转。它获取一个字符串 str 和一个格式字符串 format 。若格式字符串包含日期和时间部分，则 STR_TO_DATE() 返回一个 DATETIME 值，若该字符串只包含日期部分或时间部分,则返回一个 DATE 或 TIME 值。

mm) 函数 SUBDATE(date ,INTERVAL expr type) SUBDATE(expr ,days)

函数使用说明：当被第二个参数的 INTERVAL 型式调用时 , SUBDATE() 和 DATE_SUB() 的意义相同。对于有关 INTERVAL 参数的信息， 见有关 DATE_ADD() 的讨论。

nn) 函数 SUBTIME(expr ,expr2)

函数使用说明： SUBTIME() 从 expr 中提取 expr2 ，然后返回结果。 expr 是一个时间或日期时间表达式，而 xpr2 是一个时间表达式。

oo) 函数 SYSDATE()

函数使用说明：返回当前日期和时间值，格式为 'YYYY-MM-DD HH:MM:SS' 或 YYYYMMDDHHMMSS ， 具体格式根据函数是否用在字符串或数字语境而定。

pp) 函数 TIME(expr)

函数使用说明：提取一个时间或日期时间表达式的时间部分，并将其以字符串形式返回。

qq) 函数 TIMEDIFF(expr ,expr2)

函数使用说明： TIMEDIFF() 返回起始时间 expr 和结束时间 expr2 之间的时间。 expr 和 expr2 为时间或 date-and-time 表达式 , 两个的类型必须一样。

rr) 函数 TIMESTAMP(expr) , TIMESTAMP(expr ,expr2)

函数使用说明： 对于一个单参数 , 该函数将日期或日期时间表达式 expr 作为日期时间值返回 . 对于两个参数 , 它将时间表达式 expr2 添加到日期或日期时间表达式 expr 中，将

theresult 作为日期时间值返回。

ss) 函数 TIMESTAMPADD(interval ,int_expr ,datetime_expr)

函数使用说明：将整型表达式 int_expr 添加到日期或日期时间表达式 datetime_expr 中。int_expr 的单位被时间间隔参数给定，该参数必须是以下值的其中一个：FRAC_SECOND 、SECOND 、MINUTE 、HOUR 、DAY 、WEEK 、MONTH 、QUARTER 或 YEAR 。可使用所显示的关键词指定 Interval 值，或使用 SQL_TSI_ 前缀。例如, DAY 或 SQL_TSI_DAY 都是正确的

tt) 函数 TIMESTAMPDIFF(interval ,datetime_expr1 ,datetime_expr2)

函数使用说明：返回日期或日期时间表达式 datetime_expr1 和 datetime_expr2 the 之间的整数差。其结果的单位由 interval 参数给出。 interval 的法定值同 TIMESTAMPADD() 函数说明中所列出的相同。

uu) 函数 TIME_FORMAT(time ,format)

函数使用说明：其使用和 DATE_FORMAT() 函数相同 , 然而 format 字符串可能仅会包含处理小时、分钟和秒的格式说明符。其它说明符产生一个 NULL 值或 0 。

vv) 函数 TIME_TO_SEC(time)

函数使用说明：返回已转化为秒的 time 参数

ww) 函数 TO_DAYS(date)

函数使用说明：给定一个日期 date , 返回一个天数 (从年份 0 开始的天数) 。

xx) 函数 UNIX_TIMESTAMP(), UNIX_TIMESTAMP(date)

函数使用说明：若无参数调用，则返回一个 Unix timestamp ('1970-01-01 00:00:00' GMT 之后的秒数) 作为无符号整数。若用 date 来调用 UNIX_TIMESTAMP() ，它会将参数值以 '1970-01-01 00:00:00' GMT 后的秒数的形式返回。 date 可以是一个 DATE 字符串、一个 DATETIME 字符串、一个 TIMESTAMP 或一个当地时间的 YYMMDD 或 YYYMMDD 格式的数字。

yy) 函数 UTC_DATE, UTC_DATE()

函数使用说明：返回当前 UTC 日期值，其格式为 'YYYY-MM-DD' 或 YYYYMMDD ，具体格式取决于函数是否用在字符串或数字语境中。

zz) 函数 UTC_TIME, UTC_TIME()

函数使用说明：返回当前 UTC 值，其格式为 'HH:MM:SS' 或 HHMMSS ，具体格式根据该函数是否用在字符串或数字语境而定。

aaa) 函数 UTC_TIMESTAMP, UTC_TIMESTAMP()

函数使用说明：返回当前 UTC 日期及时间值，格式为 'YYYY-MM-DD HH:MM:SS' 或 YYYYMMDDHHMMSS ，具体格式根据该函数是否用在字符串或数字语境而定

bbb) 函数 WEEK(date [,mode])

函数使用说明：该函数返回 date 对应的星期数。 WEEK() 的双参数形式允许你指定该星期是否起始于周日或周一， 以及返回值的范围是否为从 0 到 53 或从 1 到 53 。若 mode 参数被省略，则使用 default_week_format 系统自变量的值。

ccc) 函数 WEEKDAY(date)

函数使用说明：返回 date (0 = 周一 , 1 = 周二 , ... 6 = 周日) 对应的工作日索引 weekday index for

ddd) 函数 WEEKOFYEAR(date)

函数使用说明：将该日期的阳历周以数字形式返回，范围是从 1 到 53 。它是一个兼容度函数，相当于 WEEK(date ,3) 。

eee) 函数 YEAR(date)

函数使用说明：返回 date 对应的年份 , 范围是从 1000 到 9999 。

fff) 函数 YEARWEEK(date), YEARWEEK(date ,start)

函数使用说明：返回一个日期对应的年或周。 start 参数的工作同 start 参数对 WEEK() 的工作相同。结果中的年份可以和该年的第一周和最后一周对应的日期参数有所不同。

五、 全文搜索功能函数

a) 函数 MATCH (col1,col2,...) AGAINST (expr [IN BOOLEAN MODE | WITH QUERY EXPANSION])

六、 加密函数

a) 函数 AES_ENCRYPT(str ,key_str) , AES_DECRYPT(crypt_str ,key_str)

函数使用说明：这些函数允许使用官方 AES 进行加密和数据加密 (高级加密标准) 算法 , 即以前人们所熟知的 “Rijndael” 。 保密关键字的长度为 128 比特，不过你可以通

过改变源而将其延长到 256 比特。我们选择了 128 比特的原因是它的速度要快得多，且对于大多数用途而言这个保密程度已经够用。

b) 函数 DECODE(crypt_str ,pass_str)

函数使用说明：使用 pass_str 作为密码，解密加密字符串 crypt_str ， crypt_str 应该是由 ENCODE() 返回的字符串。

c) 函数 ENCODE(str ,pass_str)

函数使用说明：使用 pass_str 作为密码，解密 str 。 使用 DECODE() 解密结果。

d) 函数 DES_DECRYPT(crypt_str [,key_str])

函数使用说明：使用 DES_ENCRYPT() 加密一个字符串。若出现错误，这个函数会返回 NULL 。

e) 函数 DES_ENCRYPT(str [,(key_num |key_str)])

函数使用说明：用 Triple-DES 算法给出的关键字加密字符串。若出现错误，这个函数会返回 NULL 。

f) 函数 ENCRYPT(str [,salt])

函数使用说明：使用 Unix crypt() 系统调用加密 str 。 salt 参数应为一个至少包含 2 个字符的字符串。若没有给出 salt 参数，则使用任意值。

g) 函数 MD5(str)

函数使用说明：为字符串算出一个 MD5 128 比特检查和。该值以 32 位十六进制数字的二进制字符串的形式返回 , 若参数为 NULL 则会返回 NULL 。例如，返回值可被用作散列关键字

h) 函数 OLD_PASSWORD(str)

函数使用说明：当 PASSWORD() 的执行变为改善安全性时， OLD_PASSWORD() 会被添加到 MySQL 。 OLD_PASSWORD() 返回从前的 PASSWORD() 执行值 (4.1 之前) ,同时允许你为任何 4.1 之前的需要连接到你的 5.1 版本 MySQL 服务器前客户端设置密码，从而不至于将它们切断

i) 函数 PASSWORD(str)

函数使用说明：从原文密码 str 计算并返回密码字符串，当参数为 NULL 时返回 NULL 。这个函数用于用户授权表的 Password 列中的加密 MySQL 密码存储

七、 信息函数

a) 函数 BENCHMARK(count ,expr)

函数使用说明： BENCHMARK() 函数重复 count 次执行表达式 expr 。 它可以被用于计算 MySQL 处理表达式的速度。结果值通常为 0 。另一种用处来自 mysql 客户端内部 ，能够报告问询执行的次数

b) 函数 CHARSET(str)

函数使用说明：返回字符串自变量的字符集。

c) 函数 COERCIBILITY(str)

函数使用说明：返回字符串自变量的整序可压缩性值。

d) 函数 COLLATION(str)

函数使用说明：返回惠字符串参数的排序方式。

e) 函数 CONNECTION_ID()

函数使用说明：返回对于连接的连接 ID (线程 ID) 。每个连接都有各自的唯一 ID 。

f) 函数 CURRENT_USER, CURRENT_USER()

函数使用说明：返回当前话路被验证的用户名和主机名组合。这个值符合确定你的存取权限的 MySQL 账户。在被指定 SQL SECURITY DEFINER 特征的存储程序内，CURRENT_USER() 返回程序的创建者

g) 函数 DATABASE()

函数使用说明：返回使用 utf8 字符集的默认 (当前) 数据库名。在存储程序里，默认数据库是同该程序向关联的数据库，但并不一定与调用语境的默认数据库相同。

h) 函数 FOUND_ROWS()

函数使用说明： A SELECT 语句可能包括一个 LIMIT 子句，用来限制服务器返回客户端的行数。在有些情况下，需要不用再次运行该语句而得知在没有 LIMIT 时到底该语句返回了多少行。为了知道这个行数 ，包括在 SELECT 语句中选择 SQL_CALC_FOUND_ROWS ，随后调用 FOUND_ROWS()

i) 函数 LAST_INSERT_ID() LAST_INSERT_ID(expr)

函数使用说明：自动返回最后一个 INSERT 或 UPDATE 问询为 AUTO_INCREMENT 列设置的第一个 发生的值。

j) 函数 ROW_COUNT()

函数使用说明： ROW_COUNT() 返回被前面语句升级的、插入的或删除的行数。 这个行数和 mysql 客户端显示的行数及 mysql_affected_rows() C API 函数返回的值相同。

k) 函数 SCHEMA()

函数使用说明：这个函数和 DATABASE() 具有相同的意义

l) 函数 SESSION_USER()

函数使用说明： SESSION_USER() 和 USER() 具有相同的意义。

m) 函数 SYSTEM_USER()

函数使用说明： SYSTEM_USER() 合 USER() 具有相同的意义

n) 函数 USER()

函数使用说明：返回当前 MySQL 用户名和机主名

o) 函数 VERSION()

函数使用说明：返回指示 MySQL 服务器版本的字符串。这个字符串使用 utf8 字符集。

八、 其他函数

a) 函数 DEFAULT(col_name)

函数使用说明：返回一个表列的默认值。若该列没有默认值则会产生错误。

b) 函数 FORMAT(X ,D)

函数使用说明：将数字 X 的格式写为 '#,###,###.##', 以四舍五入的方式保留小数点后 D 位， 并将结果以字符串的形式返回。若 D 为 0, 则返回结果不带有小数点，或不含小数部分。

c) 函数 GET_LOCK(str ,timeout)

函数使用说明：设法使用字符串 str 给定的名字得到一个锁， 超时为 timeout 秒。若成功

得到锁，则返回 1 ，若操作超时则返回 0 （ 例如 , 由于另一个客户端已提前封锁了这个名字), 若发生错误则返回 NULL (诸如缺乏记忆或线程 mysqladmin kill 被断开) 。假如你有一个用 GET_LOCK() 得到的锁，当你执行 RELEASE_LOCK() 或你的连接断开 (正常或非正常) 时，这个锁就会解除

d) 函数 INET_ATON(expr)

函数使用说明：给出一个作为字符串的网络地址的点地址表示，返回一个代表该地址数值的整数。地址可以是 4 或 8 比特地址。

e) 函数 INET_NTOA(expr)

函数使用说明：给定一个数字网络地址 (4 或 8 比特), 返回作为字符串的该地址的电地址表示

f) 函数 IS_FREE_LOCK(str)

函数使用说明：检查名为 str 的锁是否可以使用 (换言之 , 没有被封锁) 。若锁可以使用，则返回 1 (没有人在用这个锁), 若这个锁正在被使用，则返回 0 ，出现错误则返回 NULL (诸如不正确的参数) 。

g) 函数 IS_USED_LOCK(str)

函数使用说明：检查名为 str 的锁是否正在被使用 (换言之 , 被封锁) 。若被封锁，则返回使用该锁的客户端的连接标识符。否则返回 NULL 。

h) 函数 MASTER_POS_WAIT(log_name ,log_pos [,timeout])

函数使用说明：该函数对于控制主从同步很有用处。它会持续封锁，直到从设备阅读和应用主机记录中所有补充资料到指定的位置。返回值是其为到达指定位置而必须等待的记录事件的数目。若从设备 SQL 线程没有被启动、从设备主机信息尚未初始化、参数不正确或出现任何错误，则该函数返回 NULL 。若超时时间被超过，则返回 -1 。若在 MASTER_POS_WAIT() 等待期间，从设备 SQL 线程中止，则该函数返回 NULL 。若从设备由指定位置通过，则函数会立即返回结果。

i) 函数 NAME_CONST(name ,value)

函数使用说明：返回给定值。 当用来产生一个结果集合列时 , NAME_CONST() 促使该列使用给定名称。

j) 函数 RELEASE_LOCK(str)

函数使用说明：解开被 GET_LOCK() 获取的，用字符串 str 所命名的锁。若锁被解开，则返回 1 ，若改线程尚未创建锁，则返回 0 (此时锁没有被解开), 若命名的锁不存在，

则返回 NULL 。若该锁从未被对 GET_LOCK() 的调用获取，或锁已经被提前解开，则该锁不存在。

k) 函数 SLEEP(duration)

函数使用说明：睡眠 (暂停) 时间为 duration 参数给定的秒数，然后返回 0 。若 SLEEP() 被中断 , 它会返回 1 。 duration 或许或包括一个给定的以微秒为单位的分数部分。

l) 函数 UUID()

函数使用说明：返回一个通用唯一标识符 (UUID) ， UUID 被设计成一个在时间和空间上都独一无二的数字。 2 个对 UUID() 的调用应产生 2 个不同的值，即使这些调用的执行是在两个互不相连的单独电脑上进行。

m) 函数 VALUES(col_name)

函数使用说明：在一个 INSERT … ON DUPLICATE KEY UPDATE … 语句中，你可以在 UPDATE 子句中使用 VALUES(col_name) 函数，用来访问来自该语句的 INSERT 部分的列值。换言之， UPDATE 子句中的 VALUES(col_name) 访问需要被插入的 col_name 的值 , 并不会发生重复键冲突。这个函数在多行插入中特别有用。 VALUES() 函数只在 INSERT ... UPDATE 语句中有意义，而在其它情况下只会返回 NULL

九、 聚合函数

a) 函数 AVG([DISTINCT] expr)

函数使用说明：返回 expr 的平均值。 DISTINCT 选项可用于返回 expr 的不同值的平均值。

b) 函数 BIT_AND(expr)

函数使用说明：返回 expr 中所有比特的 bitwise AND 。计算执行的精确度为 64 比特 (BIGINT) 。若找不到匹配的行，则这个函数返回 18446744073709551615 。(这是无符号 BIGINT 值，所有比特被设置为 1)。

c) 函数 BIT_OR(expr)

函数使用说明：返回 expr 中所有比特的 bitwise OR 。计算执行的精确度为 64 比特 (BIGINT) 。若找不到匹配的行，则函数返回 0 。

d) 函数 BIT_XOR(expr)

函数使用说明：返回 expr 中所有比特的 bitwise XOR 。计算执行的精确度为 64 比特

(BIGINT) 。若找不到匹配的行，则函数返回 0 。

e) 函数 COUNT(expr)

函数使用说明：返回 SELECT 语句检索到的行中非 NULL 值的数目。若找不到匹配的行，则 COUNT() 返回 0

f) 函数 COUNT(DISTINCT expr ,[expr ...])

函数使用说明：返回不同的非 NULL 值数目。若找不到匹配的项，则 COUNT(DISTINCT) 返回 0

g) 函数 GROUP_CONCAT(expr)

函数使用说明：该函数返回带有来自一个组的连接的非 NULL 值的字符串结果。其完整的语法如下所示：

GROUP_CONCAT([DISTINCT] expr [,expr ...]

[ORDER BY {unsigned_integer | col_name | expr }

[ASC | DESC] [,col_name ...]]

[SEPARATOR str_val])

h) 函数 MIN([DISTINCT] expr), MAX([DISTINCT] expr)

函数使用说明：返回 expr 的最小值和最大值。 MIN() 和 MAX() 的取值可以是一个字符串参数；在这些情况下， 它们返回最小或最大字符串值。

i) 函数 STD(expr) STDDEV(expr)

函数使用说明：返回 expr 的总体标准偏差。这是标准 SQL 的延伸。这个函数的 STDDEV() 形式用来提供和 Oracle 的兼容性。可使用标准 SQL 函数 STDDEV_POP() 进行代替

j) 函数 STDDEV_POP(expr)

函数使用说明：返回 expr 的总体标准偏差(VAR_POP() 的平方根) 。你也可以使用 STD() 或 STDDEV(), 它们具有相同的意义，然而不是标准的 SQL 。若找不到匹配的行，则 STDDEV_POP() 返回 NULL

k) 函数 STDDEV_SAMP(expr)

函数使用说明：返回 expr 的样本标准差 (VAR_SAMP() 的平方根) 。若找不到匹配的行，

则 STDDEV_SAMP() 返回 NULL

l) 函数 SUM([DISTINCT] expr)

函数使用说明：返回 expr 的总数。 若返回集合中无任何行，则 SUM() 返回 NULL 。DISTINCT 关键词可用于 MySQL 5.1 中，求得 expr 不同值的总和。 若找不到匹配的行，则 SUM() 返回 NULL

m) 函数 VAR_POP(expr)

函数使用说明：返回 expr 总体标准方差。它将行视为总体，而不是一个样本， 所以它将行数作为分母。你也可以使用 VARIANCE(), 它具有相同的意义然而不是 标准的 SQL

n) 函数 VAR_SAMP(expr)

函数使用说明：返回 expr 的样本方差。更确切的说，分母的数字是行数减去 1 。若找不到匹配的行，则 VAR_SAMP() 返回 NULL

o) 函数 VARIANCE(expr)

函数使用说明：返回 expr 的总体标准方差。这是标准 SQL 的延伸。可使用标准 SQL 函数 VAR_POP() 进行代替。若找不到匹配的项，则 VARIANCE() 返回 NULL