Описание строковых функций в MySQL.

Строковые функции возвращают NULL, если длина результата оказывается больше, чем указано в значении серверного параметра max_allowed_packet.

Для функций, работающих с позициями в строке, нумерация позиций начинается с 1.

Возвращает значение ASCII-кода крайнего слева символа строки str; 0 если str является пустой строкой; NULL, если str равна NULL:

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

См. также функцию ORD().

-

ORD(str)

Если крайний слева символ строки str представляет собой многобайтный символ, то данная функция возвращает его код, который вычисляется на основе ASCII-кодов составляющих его символов по формуле: ((первый байт ASCII-кода)*256+(второй

байт ASCII-кода))[*256+третий байт ASCII-кода...]. Если крайний слева символ не является многобайтным, то данная функция возвращает то же значение, что и ASCII():

mysql> SELECT ORD('2'); -> 50

-

CONV(N,from_base,to_base)

Преобразует числа из одной системы счисления в другую. Возвращает строковое представление числа N, преобразованного из системы счисления с основой from_base в систему счисления с основой to_base. Если хотя бы один из аргументов равен NULL, то возвращается NULL. Аргумент N интерпретируется как целое число, но может быть задан как целое число или строка. Минимальное значение основы системы счисления равно 2, а максимальное - 36. Если аргумент to_base представлен отрицательным числом, то принимается, что N - число со знаком. В противном случае N трактуется как беззнаковое число. Функция CONV работает с 64-битовой точностью:

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)

Возвращает строку, представляющую двоичную величину N, где N - целое число большого размера (BIGINT). Эквивалентна функции CONV(N,10,2). Если N равно NULL, возвращается NULL:

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

-

OCT(N)

Возвращает строковое представление восьмеричного значения числа N, где N - целое число большого размера. Эквивалентно функции CONV(N,10,8). Если N равно NULL, возвращается NULL:

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

-

HEX(N_or_S)

Если N_OR_S - число, то возвращается строковое представление шестнадцатеричного числа N, где N - целое число большого размера (BIGINT). Эквивалентна функции CONV(N,10,16). Если N_OR_S - строка, то функция возвращает шестнадцатеричную строку N_OR_S, где каждый символ в N_OR_S конвертируется в 2 шестнадцатеричных числа. Является обратной по отношению к строкам 0xff.

mysql> SELECT HEX(255); -> 'FF' mysql> SELECT HEX("abc"); -> 616263 mysql> SELECT 0x616263; -> "abc"

-

CHAR(N,...)

CHAR() интерпретирует аргументы как целые числа и возвращает строку, состоящую из символов, соответствующих ASCII-коду этих чисел. Величины NULL пропускаются:

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

-

CONCAT(str1,str2,...)

Возвращает строку, являющуюся результатом конкатенации аргументов. Если хотя бы один из аргументов равен NULL, возвращается NULL. Может принимать более 2 аргументов. Числовой аргумент преобразуется в эквивалентную строковую форму:

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

-

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

Функция CONCAT_WS() обозначает CONCAT With Separator (конкатенация с разделителем) и представляет собой специальную форму функции CONCAT(). Первый аргумент является разделителем для остальных аргументов. Разделитель, так же как и остальные аргументы, может быть строкой. Если разделитель равен NULL, то результат будет NULL. Данная функция будет пропускать все величины NULL и пустые строки, расположенные после аргумента-разделителя. Разделитель будет добавляться между строками, подлежащими конкатенации:

mysql> SELECT CONCAT_WS(","," First name","Second name","Last Name"); -> 'First name,Second name,Last Name' mysql> SELECT CONCAT_WS(","," First name",NULL,"Last Name"); -> 'First name,Last Name'

-

LENGTH(str) , OCTET_LENGTH(str) , CHAR_LENGTH(str) , CHARACTER_LENGTH(str)

Возвращает длину строки str:

mysql> SELECT LENGTH('text'); -> 4 mysql> SELECT OCTET_LENGTH('text');
-> 4

Обратите внимание: для CHAR_LENGTH() и CHARACTER_LENGTH() многобайтные символы учитываются только однажды.

-

BIT_LENGTH(str)

Возвращает длину строки str в битах:

mysql> SELECT BIT_LENGTH('text'); -> 32

-

LOCATE(substr,str) , POSITION(substr IN str)

Возвращает позицию первого вхождения подстроки substr в строку str. Если подстрока substr в строке str отсутствует, возвращается 0:

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

Данная функция поддерживает многобайтные величины. В MySQL 3.23 эта функция чувствительна к регистру, а в 4.0 она чувствительна к регистру только в случае, если хотя бы один из аргументов является строкой с двоичными данными.

-

LOCATE(substr,str,pos)

Возвращает позицию первого вхождения подстроки substr в строку str, начиная с позиции pos. Если подстрока substr в строке str отсутствует, возвращается 0:

mysql> SELECT LOCATE('bar', 'foobarbar',5); -> 7

Данная функция поддерживает многобайтные величины. В MySQL 3.23 эта функция чувствительна к регистру, а в 4.0 она чувствительна к регистру, только в случае, если хотя бы один из аргументов является строкой с двоичными данными.

-

INSTR(str,substr)

Возвращает позицию первого вхождения подстроки substr в строку str. То же, что и двухаргументная форма функции LOCATE(), за исключением перемены мест аргументов:

```
mysql> SELECT INSTR('foobarbar', 'bar');          -> 4  mysql> SELECT INSTR('xbar', 'foobar');
    -> 0
```

Данная функция поддерживает многобайтные величины. В MySQL 3.23 эта функция чувствительна к регистру, а в 4.0 она чувствительна к регистру только в случае, если хотя бы один из аргументов является строкой с двоичными данными.

-

LPAD(str,len,padstr)

Возвращает строку str, которая дополняется слева строкой padstr, пока строка str не достигнет длины len символов. Если строка str длиннее, чем len, то она будет укорочена до len символов.

```
mysql> SELECT LPAD('hi',4,'??');          -> '??hi'
```

-

RPAD(str,len,padstr)

Возвращает строку str, которая дополняется справа строкой padstr, пока строка str не достигнет длины len символов. Если строка str длиннее, чем len, то она будет укорочена до len символов.

```
mysql> SELECT RPAD('hi',5,'?');          -> 'hi???'
```

-

LEFT(str,len)

Возвращает крайние слева len символов из строки str:

```
mysql> SELECT LEFT('foobarbar', 5);          -> 'fooba'
```

Данная функция поддерживает многобайтные величины.

- RIGHT(str,len)

Возвращает крайние справа len символов из строки str:

```
mysql> SELECT RIGHT('foobarbar', 4);          -> 'rbar'
```

Данная функция поддерживает многобайтные величины.

- SUBSTRING(str,pos,len) , SUBSTRING(str FROM pos FOR len) , MID(str,pos,len)

Возвращает подстроку длиной len символов из строки str, начиная от позиции pos. Существует форма с оператором FROM, для которой используется синтаксис ANSI

SQL92:

mysql> SELECT SUBSTRING('Quadratically',5,6); -> 'ratica'

Данная функция поддерживает многобайтные величины.

-

SUBSTRING(str,pos) , SUBSTRING(str FROM pos)

Возвращает подстроку из строки str, начиная с позиции pos:

mysql> SELECT SUBSTRING('Quadratically',5); -> 'ratically' mysql> SELECT SUBSTRING('foobarbar' FROM 4); -> 'barbar'

Данная функция поддерживает многобайтные величины.

-

SUBSTRING_INDEX(str,delim,count)

Возвращает подстроку из строки str перед появлениям count вхождений разделителя delim. Если count положителен, то возвращается все, что находится слева от последнего разделителя (считая слева). Если count отрицателен, то возвращается все, что находится справа от последнего разделителя (считая справа):

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

Данная функция поддерживает многобайтные величины.

-

LTRIM(str)

Возвращает строку str с удаленными начальными пробелами:

```
mysql> SELECT LTRIM(' barbar');          -> 'barbar'
```

Данная функция поддерживает многобайтные величины.

-

RTRIM(str)

Возвращает строку str с удаленными конечными пробелами:

mysql> SELECT RTRIM('barbar '); -> 'barbar'

Данная функция поддерживает многобайтные величины.

-

TRIM([[BOTH | LEADING | TRAILING] [remstr] FROM] str)

Возвращает строку str с удаленными всеми префиксами и/или суффиксами, указанными в remstr. Если не указан ни один из спецификаторов BOTH, LEADING или TRAILING, то подразумевается BOTH. Если аргумент remstr не задан, то удаляются пробелы:

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'

Данная функция поддерживает многобайтные величины.

-

SOUNDEX(str)

Возвращает саундекс от str.

Soundex - ``идентификатор звучания строки". Словосочетания ``К скалам бурым" и ``С каламбуроми" должны давать одинаковый саундекс, т.к. на слух они звучат одинаково. Заметим, однако, что этой функции для русского языка не существует. MySQL нуждается в ней, и если кто-то может предоставить алгоритм саундекса на русском языке - свяжитесь с нами. - Прим. переводчика.

Две созвучные строки, создающие впечатление почти одинаковых, могут иметь идентичные саундексы. Обычно стандартная саундекс-строка имеет длину 4 символа, но функция SOUNDEX() возвращает строку произвольной длины. Можно использовать функцию SUBSTRING() для извлечения стандартного саундекса строки из результата функции SOUNDEX(). В строке str игнорируются все символы, не являющиеся буквами или цифрами. Все международные буквенные символы, не входящие в диапазон A-Z, трактуются как гласные:

mysql> SELECT SOUNDEX('Hello'); -> 'H400' mysql> SELECT SOUNDEX('Quadratically'); -> 'Q36324'

-

SPACE(N)

Возвращает строку, состоящую из N пробелов:

```
mysql> SELECT SPACE(6);          -> ' '
```

-

REPLACE(str,from_str,to_str)

Возвращает строку str, в которой все вхождения строки from_str заменены на to_str:

```
mysql> SELECT REPLACE('www.mysql.com', 'w', 'Ww');          -> 'WwWwWw.mysql.com'
```

Данная функция поддерживает многобайтные величины.

-

REPEAT(str,count)

Возвращает строку, состоящую из строки str, повторенной count раз. Если значение count <= 0, возвращает пустую строку. Возвращает NULL, если str или count равны NULL:

```
mysql> SELECT REPEAT('MySQL', 3);          -> 'MySQLMySQLMySQL'
```

-

REVERSE(str)

Возвращает строку str с обратным порядком символов:

```
mysql> SELECT REVERSE('abc');          -> 'cba'
```

Данная функция поддерживает многобайтные величины.

-

INSERT(str,pos,len,newstr)

Возвращает строку str, в которой подстрока начиная с позиции pos, имеющая длину len замещена на newstr:

mysql> SELECT INSERT('Quadratic', 3, 4, 'What'); -> 'QuWhattic'

Данная функция поддерживает многобайтные величины.

-

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

Возвращает str1, если N = 1, str2, если N = 2, и так далее. Если N меньше, чем 1 или больше, чем число аргументов, возвращается NULL. Функция ELT() является дополненительной по отношению к функции FIELD():

mysql> SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo'); -> 'ej' mysql> SELECT ELT(4, 'ej', 'Heja', 'hej', 'foo'); -> 'foo'

-

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

Возвращает индекс строки str в списке str1, str2, str3, ... Если строка str не найдена, возвращается 0. Функция FIELD() является дополнительной по отношению к функции ELT():

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)

Возвращает значение от 1 до N, если строка str присутствует в списке strlist, состоящем из N подстрок. Список строк представляет собой строку, состоящую из подстрок, разделенных символами ','. Если первый аргумент представляет собой строку констант, а второй является столбцом типа SET, функция FIND_IN_SET() оптимизируется для использования двоичной арифметики! Возвращает 0, если str отсутствует в списке strlist или если strlist является пустой строкой. Если один из аргументов равен NULL, возвращается 0. Данная функция не будет корректно работать, если первый аргумент содержит символ ',':

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

-

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

Возвращает множество (строку, содержащую подстроки, разделенные символами ','), состоящее из строк, имеющих соответствующий бит в наборе bits. Аргумент str1 соответствует биту 0, str2 - биту 1, и так далее. Нулевые строки в наборах str1, str2, ... не прибавляются к результату:

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]])

Возвращает строку, где для каждому установленному биту в аргументе bits соответствует строка on, а каждому сброшенному биту - off. Каждая строка отделена разделителем, указанным в параметре separator (по умолчанию - ','),

причем используется только количество битов, заданное аргументом number_of_bits (по умолчанию 64), из всего количества, указанного в bits:

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

-

LCASE(str) , LOWER(str)

Возвращает строку str, в которой все символы переведены в нижний регистр в соответствии с текущей установкой набора символов (по умолчанию - ISO-8859-1 Latin1):

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

Данная функция поддерживает многобайтные величины.

-

UCASE(str) , UPPER(str)

Возвращает строку str, в которой все символы переведены в верхний регистр в соответствии с текущей установкой набора символов (по умолчанию - ISO-8859-1 Latin1):

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

Данная функция поддерживает многобайтные величины.

-

LOAD_FILE(file_name)

Читает заданный файл и возвращает его содержимое в виде строки. Данный файл должен находится на сервере, должен быть указан полный путь к этому файлу и пользователь должен обладать привилегией FILE. Размер данного файла должен быть меньше указанного в max_allowed_packet и файл должен быть открыт для чтения для всех. Если файл не существует или не может быть прочитан по одной из вышеупомянутых причин, то функция возвращает NULL:

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

При использовании версии MySQL, отличной от 3.23, чтение файла необходимо выполнять внутри вашего приложения и использовать команду INSERT для внесения в базу данных информации, содержащейся в файле. Один из путей реализации этого с использованием библиотеки MySQL++ можно найти на http://www.mysql.com/documentation/mysql++/mysql++-examples.html
.

-

QUOTE(str)

Экранирует строку с тем, чтобы получить корректное значение для SQL-выражения. Строка заключается в одинарные кавычки, и каждое вхождение одинарной кавычки ('"), обратного слеша ('\'), значения ASCII NUL и Control-Z экранируются обратным слешом. Если аргумент - NULL, то тогда результатом будет слово "NULL" без окружающих кавычек.

mysql> SELECT QUOTE("Don't"); -> 'Don't!' mysql> SELECT QUOTE(NULL); -> NULL

MySQL при необходимости автоматически конвертирует числа в строки и наоборот:

mysql> SELECT 1+"1"; -> 2 mysql> SELECT CONCAT(2,' test'); -> '2 test'

Для преобразования числа в строку явным образом, необходимо передать его в качестве аргумента функции CONCAT().

Если строковая функция содержит в качестве аргумента строку с двоичными данными, то и результирующая строка также будет строкой с двоичными данными. При этом число, конвертированное в строку, воспринимается как строка с двоичными данными. Это имеет значение только при выполнении операций сравнения.