

Funciones de cadenas

ASCII() Valor de código ASCII de un carácter

BIT_LENGTH() Cálculo de longitud de cadena en bits

CHAR() Convierte de ASCII a carácter

CHAR_LENGTH() / **CHARACTER_LENGTH()** Cálculo de longitud de cadena en caracteres

COMPRESS Comprime una cadena de caracteres

CONCAT() Concatena dos cadenas de caracteres

CONCAT_WS() Concatena cadenas con separadores

ELT() Elección entre varias cadenas

EXPORT SET() Expresiones binarias como conjuntos

FIELD() Busca el índice en listas de cadenas

FIND IN SET() Búsqueda en listas de cadenas

INSERT() Inserta una cadena en otra

INSTR() Busca una cadena en otra

LEFT() Extraer parte izquierda de una cadena

LENGTH() / **OCTET_LENGTH()** Calcula la longitud de una cadena en bytes

LOAD_FILE() Lee un fichero en una cadena

LOCATE() / **POSITION()** Encontrar la posición de una cadena dentro de otra

LOWER() / **LCASE()** Convierte una cadena a minúsculas

LPAD() Añade caracteres a la izquierda de una cadena

LTRIM() Elimina espacios a la izquierda de una cadena

MAKE SET() Crea un conjunto a partir de una expresión binaria

ORD() Obtiene el código ASCII, incluso con caracteres multibyte

QUOTE() Entrecomilla una cadena

REPEAT() Construye una cadena como una repetición de otra

REPLACE() Busca una secuencia en una cadena y la sustituye por otra

REVERSE() Invierte el orden de los caracteres de una cadena

RIGHT() Devuelve la parte derecha de una cadena

RPAD() Inserta caracteres al final de una cadena

RTRIM() Elimina caracteres blancos a la derecha de una cadena

SOUNDEX() Devuelve la cadena "soundex" para una cadena concreta

SOUNDS LIKE Compara cadenas según su pronunciación

SPACE() Devuelve cadenas consistentes en espacios

STRCMP() Compara cadenas

SUBSTRING() Extrae una subcadena de una cadena .

SUBSTRING INDEX() Extraer subcadenas de una cadena

TRIM() Elimina sufijos y/o prefijos de una cadena.

UCASE() / UPPER() Convierte una cadena a mayúsculas

UNCOMPRESS() Descomprime una cadena comprimida mediante COMPRESS

UNCOMPRESSED LENGTH() Calcula la longitud original de una cadena comprimida

Funciones de cadenas

Las funciones para tratamiento de cadenas de caracteres son:

ASCII()

ASCII(str)

Devuelve el valor de código ASCII del carácter más a la izquierda de la cadena str. Devuelve 0 si str es una cadena vacía. Devuelve NULL si str es NULL:

```
SELECT ASCII('2'), ASCII(2), ASCII('Amigo'), ASCII(''),  
ASCII(NULL);
```

	ASCII('2')	ASCII(2)	ASCII('Amigo')	ASCII('')	ASCII(NULL)
▶	50	50	65	0	NULL

Ver también la función ORD().

ORD()

ORD(str)

Si el carácter de la izquierda de la cadena str es un carácter multibyte, devuelve el código del carácter, calculado a partir de los valores de los códigos ASCII de los caracteres que lo componen, usando esta fórmula: (código ASCII del primer byte)*256+(código ASCII del segundo byte)[*256+ código ASCII del tercer byte...]. Si el carácter no es multibyte, devuelve el mismo valor que la función ASCII():

```
SELECT ORD('2'), ORD('Amigo'), ORD('
```

	ORD('2')	ORD('Amigo')	ORD('
▶	50	65	13

CHAR()

CHAR(N,...)

CHAR() interpreta los argumentos como enteros y devuelve una cadena que consiste en los caracteres dados por los valores de los códigos ASCII de esos enteros. Los valores NULL se saltan:

```
SELECT CHAR(77,121,83,81,'76'), CHAR(77,77.3,'77.3');
```

	CHAR(77,121,83,81,'76')	CHAR(77,77.3,'77.3')
▶ MySQL	MMM	MMM

LENGTH() / OCTET LENGTH()

LENGTH(str)

OCTET_LENGTH(str)

Devuelve la longitud de la cadena str, medida en bytes. Un carácter multibyte cuenta cómo bytes múltiples. Esto significa que para cadenas que contengan cinco caracteres de dos bytes, **LENGTH()** devuelve 10, mientras que **CHAR_LENGTH()** devuelve 5.

```
SELECT LENGTH('text'), LENGTH('Calendario'), LENGTH('H  
o  
l  
a');
```

	LENGTH('text')	LENGTH('Cale...')	LENGTH('H
▶	4	10	13

OCTET_LENGTH() es sinónimo de **LENGTH()**.

BIT LENGTH()

BIT_LENGTH(str)

Devuelve la longitud de la cadena str en bits:

```
S SELECT BIT_LENGTH('text'), BIT_LENGTH('Calendario');ELECT  
BIT_LENGTH('text');
```

	BIT_LENGTH('text')	BIT_LENGTH('Calendario')
▶	32	80

CHAR LENGTH() / CHARACTER LENGTH()

CHAR_LENGTH(str)

CHARACTER_LENGTH(str)

Devuelve la longitud de la cadena str, medida en caracteres. Un carácter multibyte cuenta como un carácter sencillo. Esto significa que para una cadena que contenga cinco caracteres de dos bytes, **LENGTH()** devuelve 10, mientras que **CHAR_LENGTH()** devuelve 5.

```
SELECT CHAR_LENGTH('text'), CHAR_LENGTH('Calendario'), CHAR_LENGTH('H  
o  
l  
a');
```

CHAR_LENGTH...	CHAR_LENGTH...	CHAR_LENGTH...
4	10	13

COMPRESS

COMPRESS(string_to_compress)

Comprime una cadena.

```
SELECT LENGTH(COMPRESS(REPEAT("a",1000))), LENGTH(COMPRESS("")),  
COMPRESS("a"), LENGTH(COMPRESS("a")), COMPRESS(REPEAT("a",16));
```

LENGTH(COMPRESS(REPEAT("a",1000)))	LENGTH(COMPRESS(""))	COMPRESS("a")	LENGTH(COMPRESS("a"))	COMPRESS(REPEAT("a",16))
21	0	xKz b	13	+ xKLD 3

COMPRESS() fue añadido en MySQL 4.1.1. Requiere que MySQL se haya compilado con una librería de compresión como **zlib**. De otro modo, el valor de retorno es siempre NULL. El contenido de la cadena comprimida se almacena del siguiente modo:

- Las cadenas vacías se almacenan como cadenas vacías.
- Las cadenas no vacías se almacenan como un dato de 4 bytes con la longitud de la cadena sin comprimir (el byte de menor peso primero), seguido por la cadena comprimida usando gzip. Si la cadena termina con un espacio, se añade un '.' extra para impedir problemas con la posible eliminación de espacios si el resultado se almacena en una columna *CHAR* o *VARCHAR*. El uso de *CHAR* o *VARCHAR* para almacenar cadenas comprimidas no se recomienda. Es mucho mejor usar una columna *BLOB* en su lugar.

UNCOMPRESS

UNCOMPRESS(string_to_uncompress)

Descomprime una cadena comprimida con la función COMPRESS().

```
SELECT COMPRESS("any string"), UNCOMPRESS(COMPRESS("any  
string"));
```

COMPRESS("any string")	UNCOMPRESS(COMPRESS("any string"))
xKz b	BLOB

UNCOMPRESS() Requiere que MySQL haya sido compilado con una librería de compresión como **zlib**. De otro modo, el valor de retorno es siempre NULL.

UNCOMPRESSED LENGTH

UNCOMPRESSED_LENGTH(compressed_string)

Devuelve la longitud de una cadena comprimida antes de su compresión.

```
SELECT LENGTH(COMPRESS("any string")),  
UNCOMPRESSED_LENGTH(COMPRESS("any string"));
```

LENGTH(COMPRESS("any string"))	UNCOMPRESSED_LENGTH(COMPRESS("any string"))
22	10

CONCAT

CONCAT(str1, str2,...)

Devuelve la cadena resultante de concatenar los argumentos. Devuelve NULL si alguno de los argumentos es NULL. Puede haber más de 2 argumentos. Un argumento numérico se convierte a su cadena equivalente:

```
SELECT CONCAT('My', 'S', 'QL'), CONCAT('My', NULL, 'QL'),  
CONCAT(14.3);
```

CONCAT('My','S','QL')	CONCAT('My',NULL,'QL')	CONCAT(14.3)
MySQL	NULL	14.3

```
SELECT EmNombre, CONCAT(EmSalario, '+', IFNULL(EmComision, 0))  
FROM Empleados
```

EmNombre	CONCAT(EmSalario, '+', IFNULL(EmComision, 0))
PONS, CESAR	3100+0
LASA, MARIO	3500+1100
TEROL, LUCIANO	2900+1100
PEREZ, JULIO	4400+0
AGUIRRE, AUREO	3100+1100
PEREZ, MARCOS	4800+500
VEIGA, JULIANA	3000+0
GÁLVEZ, PILAR	3800+0
SANZ, LAVINIA	2800+1000
ALBA, ADRIANA	4500+0
LOPEZ, ANTONIO	7200+0
GARCIA, OCTAVIO	3800+800
FLOR, DOROTEA	2900+800
POLO, OTILIA	3800+0

34 rows fetched in 0,0123s (0,0006s)

CONCAT_WS

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

CONCAT_WS() funciona como CONCAT() pero con separadores y es una forma especial de CONCAT(). El primer argumento es el separador para el resto de los argumentos. El separador se añade entre las cadenas a concatenar: El separador puede ser una cadena, igual que el resto de los argumentos. Si el separador es NULL,

el resultado es NULL. La función pasa por alto cualquier valor NULL después del argumento separador.

```
SELECT CONCAT_WS(",", "First name", "Second name", "Last Name"),
CONCAT_WS(",", "First name", NULL, "Last Name");
```

CONCAT_WS("","First name","Second name","Last Name")	CONCAT_WS("","First name",NULL,"Last Name")
First name,Second name,Last Name	First name,Last Name

ELTQ

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

Devuelve str1 si N = 1, str2 si N = 2, y sucesivamente. Devuelve NULL si N es menor que 1 o mayor que el número de argumentos. **ELT()** es el complemento de **FIELD()**:

```
SELECT ELT(1, 'ej', 'Heja', 'hej', 'foo'), ELT(4, 'ej', 'Heja', 'hej', 'foo');
```

ELT(1, 'ej', 'Heja', 'hej', 'foo')	ELT(4, 'ej', 'Heja', 'hej', 'foo')
ej	foo

```
SELECT EmNombre, EmNumHijos, ELT(EmNumHijos, EmNombre,
EmSalario, EmComision, EmCodigoDepartamento, EmExTelefono) FROM
Empleados;
```

EmNombre	EmNumHijos	ELT(EmNumHijos, EmNombre, EmSalario, EmComision, Em...
PONS, CESAR	3	NULL
LASA, MARIO	1	LASA, MARIO
TEROL, LUCIANO	2	2900
PEREZ, JULIO	0	NULL
AGUIRRE, AUREO	2	3100
PEREZ, MARCOS	2	4800
VEIGA, JULIANA	4	121
GALVEZ, PILAR	2	3800
SANZ, LAVINIA	3	1000
ALBA, ADRIANA	0	NULL
LOPEZ, ANTONIO	6	NULL
GARCIA, OCTAVIO	3	800
FLOR, DOROTEA	5	410
POLO, OTILIA	0	NULL

FIELDQ

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

Devuelve el índice de 'str' en la lista 'str1', 'str2', 'str3', Devuelve 0 si 'str' no se encuentra. **FIELD()** es el complemento de **ELD()**

```
SELECT FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo'),
FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo');
```

FIELD('ej', 'Hej', 'ej', 'Heja', 'hej', 'foo')	FIELD('fo', 'Hej', 'ej', 'Heja', 'hej', 'foo')
2	0

EXPORT SET()

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

Devuelve una cadena donde para todos los bits activos en 'bits', se obtiene una cadena 'on' y para los inactivos se obtiene una cadena 'off'. Cada cadena se separa con 'separator' (por defecto ',') y sólo 'number_of_bits' (por defecto 64) de 'bits' se usan:

```
SELECT EXPORT_SET(5, 'Y', 'N', ',', 4), EXPORT_SET(7, 'Si', 'No', '*', 6), EXPORT_SET(7, 'Si', 'No', '*', 3);
```

EXPORT_SET(5,'Y','N',',',4)	EXPORT_SET(7,'Si','No','*',6)	EXPORT_SET(7,'Si','No','*',3)
Y,N,Y,N	Si*Si*Si*No*No*No	Si*Si*Si

FIND IN SET()

FIND_IN_SET(str, strlist)

Devuelve un valor de 1 a N si la cadena str está en la lista strlist que consiste en N subcadenas. Una lista de cadenas es una cadena compuesta por subcadenas separadas por caracteres ','. Si el primer argumento es una cadena constante y la segunda es una columna de tipo SET, la función FIND_IN_SET() está optimizada para usar aritmética de bits. Devuelve 0 si str no está en strlist o si strlist es una cadena vacía. Devuelve NULL si cualquiera de los argumentos es NULL. Esta función no funcionará adecuadamente si el primer argumento contiene una coma ',':

```
SELECT FIND_IN_SET('b', 'a,b,c,d');
```

FIND_IN_SET('b','a,b,c,d')
2

INSERT()

INSERT(str, pos, len, newstr)

Devuelve la cadena str, con la subcadena que empieza en la posición pos y de len caracteres de longitud remplazada con la cadena newstr:

```
SELECT INSERT('Quadratic', 3, 2, 'What'), INSERT('Quadratic', 3, 4, 'Whathon');
```


INSERT('Quadratic', 3, 2, 'What')	INSERT('Quadratic', 3, 4, 'Whathon')
QuWWhatratic	QuWWhathontic

Esta función es segura con caracteres multibyte.

```
SELECT INSERT(DeNombre, 5, 2, 'DEPARTAMENTO') FROM Departamentos
;
```

INSERT(DeNombre, 5, 2, 'DEPARTAMENTO')
CONTDEPARTAMENTOILIDAD
DIREDEPARTAMENTOION COMERCIAL
DIREDEPARTAMENTOION GENERAL
FINADEPARTAMENTOAS
ORGADEPARTAMENTOZACION
PERSDEPARTAMENTOAL
PERSDEPARTAMENTOAL CONTRATADO
PROCDEPARTAMENTO DE DATOS
PRODDEPARTAMENTOION
SECTDEPARTAMENTO INDUSTRIAL
SECTDEPARTAMENTO SERVICIOS

INSTR

INSTR(str, substr)

Devuelve la posición de la primera aparición de la subcadena substr dentro de la cadena str. Es lo mismo que la forma de LOCATE() con dos argumentos, excepto que los argumentos están intercambiados:

```
SELECT INSTR('foobarbar', 'bar'), INSTR('xbar', 'foobar');
```

INSTR('foobarb...	INSTR('xbar', 'foobar')
4	0

```
SELECT EmNombre, INSTR(EmNombre, 'ez') FROM Empleados
WHERE INSTR(EmNombre, 'ez') <> 0;)
```

EmNombre	INSTR(EmNom...
DIEZ, AMELIA	3
GALVEZ, PILAR	5
LOPEZ, ANTONIO	4
PEREZ, JULIO	4
PEREZ, MARCOS	4
PEREZ, SABINA	4
VAZQUEZ, HONORIA	6

Esta función es segura con caracteres multibyte. En MySQL 3.23 esta función distingue mayúsculas y minúsculas, mientras que en la versión 4.0 no lo hace si cualquiera de los argumentos es una cadena binaria.

LOCATE() / POSITION()

LOCATE(substr, str)

LOCATE(substr, str, pos)

POSITION(substr IN str)

La primera forma devuelve la posición de la primera aparición de la cadena substr dentro de la cadena str. La segunda devuelve la posición de la primera aparición de la cadena substr dentro de la cadena str, comenzando en la posición pos. Devuelve 0 si substr no está en str.

```
SELECT LOCATE('bar', 'foobarbar'), LOCATE('xbar', 'foobar'),  
LOCATE('bar', 'foobarbar', 5);
```

LOCATE('bar', 'foobarbar')	LOCATE('xbar', 'foobar')	LOCATE('bar', 'foobarbar', 5)
4	0	7

Esta función es segura con cadenas multibyte. En MySQL 3.23 esta función distingue entre mayúsculas y minúsculas, en la versión 4.0 sólo si cualquiera de los argumentos es una cadena binaria.

```
SELECT EmNombre, LOCATE('Juli', EmNombre) FROM Empleados  
WHERE LOCATE('Juli', EmNombre) <> 0;
```

EmNombre	LOCATE('Juli', ...)
PEREZ, JULIO	8
VEIGA, JULIANA	8

```
SELECT DeNombre, LOCATE('CO', DeNombre) FROM Departamentos  
WHERE LOCATE('CO', DeNombre) <> 0;
```

DeNombre	LOCATE('CO', ...)
CONTABILIDAD	1
DIRECCION COMERCIAL	11
PERSONAL CONTRATADO	10

```
SELECT DeNombre, LOCATE('CO', DeNombre, 6) FROM Departamentos  
WHERE LOCATE('CO', DeNombre, 6) <> 0;
```

DeNombre	LOCATE('CO', ...)
DIRECCION COMERCIAL	11
PERSONAL CONTRATADO	10

POSITION(substr IN str) es sinónimo de **LOCATE(substr, str)**.

```
SELECT DeNombre, POSITION('CO' IN DeNombre) FROM Departamentos
WHERE POSITION('CO' IN DeNombre) <> 0;
```

DeNombre	POSITION('CO' IN DeNombre)
▶ CONTABILIDAD	1
DIRECCION COMERCIAL	11
PERSONAL CONTRATADO	10

LOWER() / LCASE()

LOWER(str)

LCASE(str)

Devuelve la cadena str con todos los caracteres cambiados a minúsculas de acuerdo con el mapa de caracteres actual (por defecto es ISO-8859-1 Latin1):

```
SELECT LOWER('QUADRATICALLY'), LCASE('QUADRATICALLY');
```

LOWER('QUADRATICALLY')	LCASE('QUADRATICALLY')
▶ quadratically	quadratically

LCASE() es sinónimo de **LOWER()**.

Esta función es segura para caracteres multibyte.

UCASE() / UPPER()

UCASE(str)

UPPER(str)

Devuelve la cadena str con todos sus caracteres sustituidos a mayúsculas de acuerdo con el mapa del conjunto de caracteres actual (por defecto es ISO-8859-1 Latin1):

```
SELECT UPPER('Hej'), UCASE('hola');
```

UPPER('Hej')	UCASE('hola')
▶ HEJ	HOLA

Esta función es segura "multi-byte".

UCASE() es sinónimo de **UPPER()**.

LEFT()

LEFT(cadena, longitud)

Devuelve los 'longitud' caracteres de la izquierda de la 'cadena':

```
SELECT LEFT('MySQL, Curso Profesional', 5);
```

LEFT('MySQL, Curso Pr...	
▶ MySQL	

RIGHTQ

RIGHT(cadena, longitud)

Devuelve los 'longitud' caracteres de la derecha de la 'cadena':

```
SELECT RIGHT('MySQL, Curso Profesional', 18);
```

RIGHT('MySQL, Curso Profesional', 18)	
▶ Curso Profesional	

Esta función es segura "multi-byte".

LTRIMQ

LTRIM(str)

Devuelve la cadena str con los caracteres de espacios iniciales eliminados:

```
SELECT LTRIM('          barbar');
```

LTRIM(' barbar')	
▶ barbar	

RTRIMQ

RTRIM(str)

Devuelve la cadena str con los caracteres de espacios finales eliminados:

```
SELECT RTRIM('barbar          ');
```

RTRIM('barbar ')	
▶ barbar	

Esta función es segura con caracteres multibyte.

TRIMQ

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

Devuelve la cadena str eliminando todos los prefijos y/o sufijos remstr. Si no se incluye ninguno de los especificadores BOTH(inicio y final del campo), LEADING(inicio del

campo) o TRAILING(final del campo), se asume BOTH. Si no se especifica la cadena remstr, se eliminan los espacios:

```
SELECT TRIM(LEADING 'x' FROM 'xxxbarxxx'), TRIM(BOTH 'x' FROM 'xxxbarxxx'), TRIM(TRAILING 'xyz' FROM 'barxyz');
```

TRIM(LEADING 'x' FROM 'xxxbarxxx')	TRIM(BOTH 'x' FROM 'xxxbarxxx')	TRIM(TRAILING 'xyz' FROM 'barx...')
barxxx	bar	barx

Esta función es segura "multi-byte".

```
SELECT EmNombre, TRIM(LEADING 'PE' FROM EmNombre) FROM Empleados WHERE EmNombre LIKE 'P%';
```

EmNombre	TRIM(LEADING 'PE' FROM EmNombre)
PEREZ, JULIO	REZ, JULIO
PEREZ, MARCOS	REZ, MARCOS
PEREZ, SABINA	REZ, SABINA
PINO, DIANA	PINO, DIANA
POLO, OTILIA	POLO, OTILIA
PONS, CESAR	PONS, CESAR

SUBSTRING

SUBSTRING(str, pos, car)

Devuelve una parte de una cadena de caracteres. Str puede ser una cadena o el nombre de una columna, pos es la posición inicial si no se indica, se asume desde la primera posición, car es el número de caracteres que se devuelven.

```
SELECT SUBSTRING('www.mysql.com', 5, 6), SUBSTRING('www.mysql.com', 6);
```

SUBSTRING('www.mysql.co...')	SUBSTRING('www.mysql.com', 6)
mysql.	ysql.com

```
SELECT EmNombre, SUBSTRING(EmNombre, 7) FROM Empleados;
```

EmNombre	SUBSTRING(EmNombre, 7)
AGUIRRE, AUREO	E, AUREO
ALBA, ADRIANA	ADRIANA
CAMPOS, ROMULO	, ROMULO
CAMPS, AURELIO	AURELIO
DIEZ, AMELIA	AMELIA
DURAN, LIVIA	LIVIA
FIERRO, CLAUDIA	, CLAUDIA
FLOR, DOROTEA	DOROTEA
GALVEZ, PILAR	, PILAR
GARCIA, AUGUSTO	, AUGUSTO
GARCIA, OCTAVIO	, OCTAVIO
GIL, GLORIA	LORIA
LARA, DORINDA	DORINDA
LARA, LUCRECIA	LUCRECIA

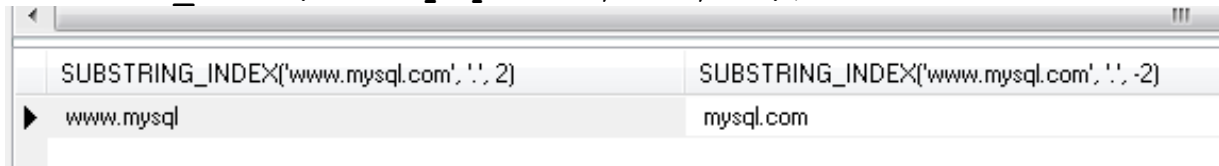
34 rows fetched in 0.0031s (0.0004s)

SUBSTRING_INDEX

SUBSTRING_INDEX(str, delim, count)

Devuelve la subcadena de str anterior a la aparición de count veces el delimitador delim. Si count es positivo, se retorna todo lo que haya a la izquierda del delimitador final (contando desde la izquierda). Si count es negativo, se devuelve todo lo que haya a la derecha del delimitador final (contando desde la derecha):

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



SUBSTRING_INDEX('www.mysql.com', '.', 2)	SUBSTRING_INDEX('www.mysql.com', '.', -2)
www.mysql	mysql.com

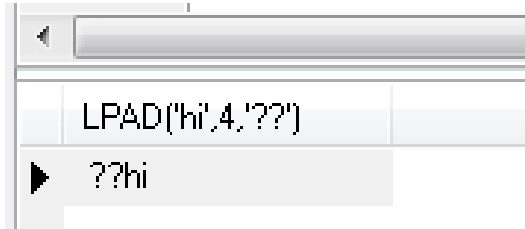
Esta función es segura "multi-byte".

LPAD

LPAD(str, len, padstr)

Devuelve la cadena str, rellena a la izquierda con la cadena padstr hasta la longitud de len caracteres. Si str es más larga que len, el valor retornado se acorta hasta len caracteres.

```
SELECT LPAD('hi', 4, '??');
```



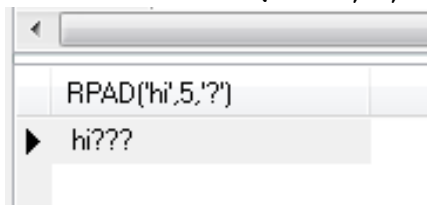
LPAD('hi', 4, '??')
??hi

RPAD

RPAD(str, len, padstr)

Devuelve la cadena str, rellena a la derecha con la cadena padstr hasta la longitud de len caracteres. Si str es más larga que len, el valor retornado se acorta hasta len caracteres.

```
SELECT RPAD('hi', 5, '?');
```



RPAD('hi', 5, '?')
hi???

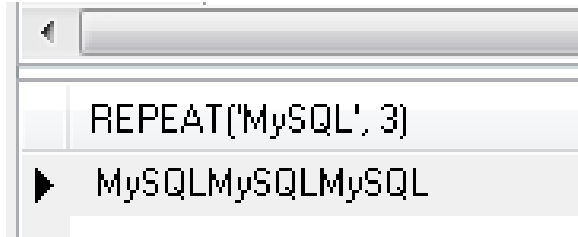
Esta función es segura con caracteres multibyte.

REPEAT()

REPEAT(str, count)

Devuelve una cadena que consiste en la cadena str repetida count veces. Si count <= 0, devuelve una cadena vacía. Devuelve NULL si str o count son NULL:

```
SELECT REPEAT('MySQL', 3);
```



A screenshot of a SQL query result. The query is `REPEAT('MySQL', 3)`. The result is displayed in a table with one row and one column, showing the output `MySQLMySQLMySQL`.

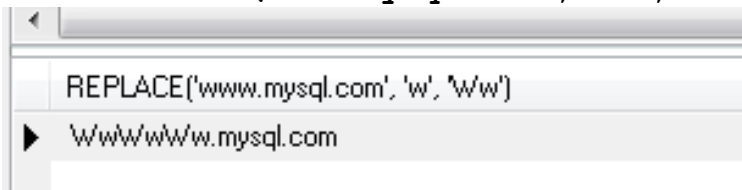
REPEAT('MySQL', 3)
MySQLMySQLMySQL

REPLACE()

REPLACE(str, from_str, to_str)

Devuelve la cadena str con todas las apariciones de la cadena from_str sustituidas por la cadena to_str:

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



A screenshot of a SQL query result. The query is `REPLACE('www.mysql.com', 'w', 'Ww')`. The result is displayed in a table with one row and one column, showing the output `WwWwWw.mysql.com`.

REPLACE('www.mysql.com', 'w', 'Ww')
WwWwWw.mysql.com

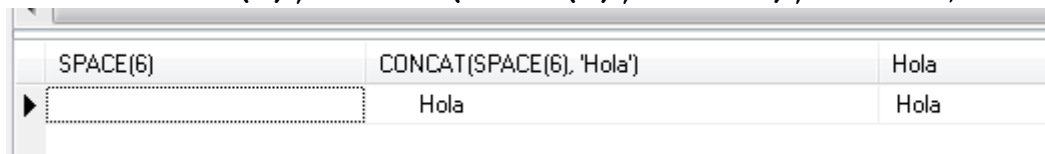
Esta función es segura a para caracteres multibyte.

SPACE()

SPACE(N)

Devuelve una cadena que consiste en N caracteres espacio:

```
SELECT SPACE(6), CONCAT(SPACE(6), 'Hola'), 'Hola';
```



A screenshot of a SQL query result. The query is `SPACE(6), CONCAT(SPACE(6), 'Hola'), 'Hola'`. The result is displayed in a table with three columns. The first column is `SPACE(6)`, the second is `CONCAT(SPACE(6), 'Hola')`, and the third is `Hola`. The first row shows the output of the functions, and the second row shows the output of the concatenation.

SPACE(6)	CONCAT(SPACE(6), 'Hola')	Hola
	Hola	Hola

QUOTE()

QUOTE(str)

Entrecomilla una cadena para producir un resultado que se pueda utilizar correctamente como valor escapado en una declaración de los datos SQL. Se devuelve la cadena entre apóstrofes y con cada aparición del carácter '\', ASCII NUL, apóstrofe (''), y el Control-Z precedido por un '\'. Si el argumento es NULL, el valor devuelto es la palabra 'NULL' sin las comillas.

```
SELECT QUOTE('Don't'), QUOTE(NULL);
```

QUOTE('Don't')	QUOTE(NULL)
'Don\'t'	NULL

REVERSE()

REVERSE(str)

Devuelve la cadena str con el orden de los caracteres invertido:

```
SELECT REVERSE('abc');
```

REVERSE('a...')
cba

Esta función es segura con caracteres multibyte.

```
SELECT EmNombre, REVERSE(EmNombre) FROM empleados
```

EmNombre	REVERSE(EmNombre)
AGUIRRE, AUREO	DERUA ,ERRIUGA
ALBA, ADRIANA	ANAIROA ,ABLA
CAMPOS, ROMULO	OLUMOR ,SOPMAC
CAMPS, AURELIO	OILERUA ,SPMAC
DIEZ, AMELIA	AILEMA ,ZEID
DURAN, LIVIA	AIVIL ,NARUD
FIERRO, CLAUDIA	AIDUALC ,ORREIF
FLOR, DOROTEA	AETOROD ,ROLF
GALVEZ, PILAR	RALIP ,ZEVLAG
GARCIA, AUGUSTO	OTSUGUA ,AICRAG
GARCIA, OCTAVIO	OIVATCO ,AICRAG
GIL, GLORIA	AIROLG ,LIG
LARA, DORINDA	ADNIROD ,ARAL
LARA, LUCRECIA	AICERCUL ,ARAL

34 rows fetched in 0,0047s (0,0005s)

```
select IF('ama' = REVERSE('amar'), 'palindroma', 'distintas');
```

MAKE_SET()

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

Devuelve un conjunto (una cadena que contiene subcadenas separadas por caracteres ',') consistente en las cadenas correspondientes a los bits activos en bits. str1 corresponde al bit 0, str2 al bit 1, etc. Las cadenas NULL en str1, str2, ... no se añaden al resultado:

```
SELECT MAKE_SET(1, 'a', 'b', 'c'), MAKE_SET(1 | 7, 'hello', 'nice', 'world'), MAKE_SET(0, 'a', 'b', 'c');
```

MAKE_SET(1, 'a', 'b', 'c')	MAKE_SET(1 7, 'hello', 'nice', 'world')	MAKE_SET(0, 'a', 'b', 'c')
a	hello,nice,world	

SOUNDEX()

SOUNDEX(str)

Devuelve una cadena 'soundex' desde str. Dos cadenas que suenen casi igual tienen cadenas soundex idénticas. Una cadena soundex estándar tiene 4 caracteres de longitud, pero **SOUNDEX()** devuelve una cadena de longitud arbitraria. Se puede usar SUBSTRING() sobre el resultado para obtener una cadena soundex estándar. Cualquier carácter no alfanumérico será ignorado. Todos los caracteres alfabéticos fuera del rango A-Z serán tratados como vocales:

```
SELECT SOUNDEX('Hello'), SOUNDEX('Hola'), SOUNDEX('Cuadrado'),  
SOUNDEX('Cuadro');
```

SOUNDEX('Hello')	SOUNDEX('Hola')	SOUNDEX('Cuadrado')	SOUNDEX('Cuadro')
H400	H400	C363	C360

SOUNDS LIKE

expr1 SOUNDS LIKE expr2

Es lo mismo que hacer la comparación SOUNDEX(expr1) = SOUNDEX(expr2). Está disponible sólo a partir de MySQL 4.1.

```
SELECT IF(SOUNDEX('Hello') = SOUNDEX('Hola'), 'Iguales', 'No'),  
IF(SOUNDEX('Cuadrado') = SOUNDEX('Rectangulo'), 'Iguales',  
'No');
```

IF(SOUNDEX('Hello') = SOUNDEX('Hola'), 'Iguales', 'No')	IF(SOUNDEX('Cuadrado') = SOUNDEX('Rectangulo'), 'Iguales', 'No')
Iguales	No

expr1 SOUNDS LIKE expr2 Es equivalente a:
SOUNDEX(expr1)=SOUNDEX(expr2)

```
SELECT IF('Hello' SOUNDS LIKE 'Hola', 'Iguales', 'No'),  
IF('Cuadrado' SOUNDS LIKE 'Rectangulo', 'Iguales', 'No');
```

IF('Hello' SOUNDS LIKE 'Hola', 'Iguales', 'No')	IF('Cuadrado' SOUNDS LIKE 'Rectangulo', 'Iguales', 'No')
Iguales	No

STRCMP

STRCMP(expr1,expr2)

STRCMP() devuelve 0 si las cadenas son iguales, -1 si el primer argumento es menor que el segundo, según el orden de ordenamiento de cadenas actual, y 1 en otro caso.

```
SELECT STRCMP('text', 'text2'), STRCMP('text2', 'text'),  
STRCMP('text', 'text');
```

STRCMP('text', 'text2')	STRCMP('text2', 'text')	STRCMP('text', 'text')
-1	1	0

Desde MySQL 4.0, **STRCMP()** usa el conjunto de caracteres actual cuando realiza las comparaciones. Esto hace que el comportamiento de comparación por defecto no sea sensible al tipo, a no ser que uno o ambos operandos sean cadenas binarias.

LOAD_FILE()

LOAD_FILE(file_name)

Lee el fichero y devuelve su contenido como una cadena. El fichero debe estar en el servidor, se debe especificar el camino completo al fichero, y se debe poseer el privilegio FILE. El fichero debe ser legible para todos y más pequeño que max_allowed_packet. Si el fichero no existe o no puede ser leído por alguna de las razones anteriores, la función devuelve NULL:

```
LOAD DATA LOCAL INFILE 'Datos.txt' INTO TABLE Prueba;
UPDATE tbl_name
SET blob_column=LOAD_FILE("/tmp/picture")
WHERE id=1;
```