

1.1.1 Funcións incorporadas en MySQL

Cando se necesita facer algunha operación complexa, débese de consultar o manual para saber se xa existe algunha función implícita no servidor para resolvela antes de poñerse a crear unha función nova de usuario. MySQL incorpora unha gran cantidade de funcións que poden ser utilizadas nas expresións contidas nas consultas. O manual de referencia de MySQL posúe información moi detallada das funcións; aquí móstrase un resumo das máis utilizadas.

1.1.1.1 Funcións de data e hora

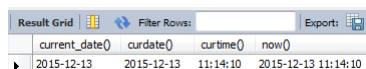
Permiten obter información sobre a data e hora do sistema en distintos formatos, resolver os problemas de aritmética de datas (cálculos aritméticos feitos con datos tipo data e hora) e cambiar o formato de saída.

Información sobre data e hora do servidor

- **CURRENT_DATE() ou CURDATE()**
Devolve a data actual do sistema, en formato 'aaaa-mm-dd'
- **CURRENT_TIME() ou CURTIME()**
Devolve a hora actual do sistema, en formato 'hh:mm:ss'
- **NOW()**
Devolve a data e hora do sistema, en formato 'aaaa-mm-dd hh:mm:ss'

Exemplo considerando que agora son as 11:14:10 do 13/12/2015:

```
select current_date(), curdate(), curtime(), now();
```



current_date()	curdate()	curtime()	now()
2015-12-13	2015-12-13	11:14:10	2015-12-13 11:14:10

Aritmética de datas

Para facer operacións aritméticas con datas en MySQL, hai que utilizar as funcións que incorpora para tal fin. Algunhas destas funcións poden manexar intervalos de tempo facendo referencia ás palabras reservadas: MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, ou YEAR.

- **DATE_ADD**
Suma (*add*) a unha data que se pasa como primeiro parámetro o intervalo de tempo especificado como segundo parámetro. Sintaxe:

```
DATE_ADD(data, INTERVAL expresión unidade_intervalo)
```

- **ADDDATE**
Suma (*add*), a unha data que se pasa como primeiro parámetro, un número de días ou un intervalo de tempo, que se pasan como segundo parámetro. Sintaxe:

```
ADDDATE(data, {número_días | INTERVAL expresión unidade_intervalo })
```

Cando se utiliza coa forma INTERVAL no segundo parámetro, dá o mesmo resultado cá función DATE_ADD.

- **DATE_SUB**

Resta (*subtract*) a unha data que se pasa como primeiro parámetro o intervalo de tempo especificado como segundo parámetro. Sintaxe:

```
DATE_ADD(data, INTERVAL expresión unidade_intervalo)
```

- **SUBDATE**

Resta (*subtract*), a unha data que se pasa como primeiro parámetro, un número de días ou un intervalo de tempo especificado como segundo parámetro. Sintaxe:

```
SUBDATE(data, {número_días|INTERVAL expresión unidade_intervalo})
```

Cando se utiliza coa forma INTERVAL no segundo parámetro, dá o mesmo resultado cá función DATE_SUB.

- **DATEDIFF**

Devolve o número de días transcorridos entre dúas datas que se pasan como parámetros. Sintaxe:

```
DATEDIFF(data_maior, data_menor)
```

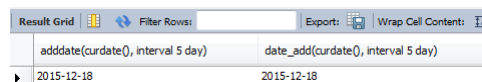
- **TIMESTAMPDIFF**

Devolve o número de intervalos de tempo (MICROSECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, ou YEAR) que se pasa como primeiro parámetro, transcorridos entre dúas expresións tipo data e hora, ou tipo data. Sintaxe:

```
TIMESTAMPDIFF(unidade_intervalo, data_hora_inicio, data_hora_fin)
```

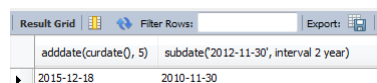
Exemplos considerando que hoxe é 2015/12/13 (según o formato de MySQL):

```
select adddate(curdate(), interval 5 day), date_add(curdate(), interval 5 day);
```



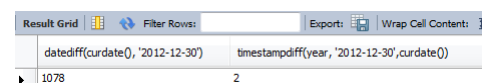
adddate(curdate(), interval 5 day)	date_add(curdate(), interval 5 day)
2015-12-18	2015-12-18

```
select adddate(curdate(), 5), subdate('2012-11-30', interval 2 year);
```



adddate(curdate(), 5)	subdate('2012-11-30', interval 2 year)
2015-12-18	2010-11-30

```
select datediff(curdate(), '2012-12-30'), timestampdiff(year, '2012-12-30', curdate());
```



datediff(curdate(), '2012-12-30')	timestampdiff(year, '2012-12-30', curdate())
1078	2

Formato de saída para datos tipo data e hora

- **DATE_FORMAT**

Aplica un formato de saída a unha expresión tipo data, ou tipo data e hora. Sintaxe:

```
DATE_FORMAT(data, 'cadea_de_formato')
```

A cadea de formato representa a forma en que vai a mostrar a data. Dentro da cadea pódese poñer calquera carácter que será mostrado tal e como aparece na cadea de formato, e ademais, pode levar o carácter % que engade un significado especial a algúns caracteres. Algunhas combinacións de caracteres especiais permitidas son:

%d	Día do mes, con dous díxitos (entre 01 e 31)
%a	Abreviatura do nome do día da semana (en inglés: Sun...Sat)
%W	Nome do día da semana (en inglés: Sunday...Saturday)
%w	Día da semana en cifras (0 = domingo ... 6 = sábado)
%m	Mes, con dous díxitos (entre 01 e 12)
%b	Abreviatura do nome do mes (en inglés: Jan...Dec)
%M	Nome do mes (en inglés: January...December)
%y	Ano, con dous díxitos
%Y	Ano, con catro díxitos
%h	Hora, con dous díxitos (entre 01 e 12)
%H	Hora, con dous díxitos (entre 00 e 23)
%i	Minutos, con dous díxitos (entre 00 e 59)
%s	Segundos, con dous díxitos (entre 00 e 59)
%r	Hora con formato 12 horas (hh:mm:ss), seguido de AM ou PM
%p	AM ou PM
%%	Mostra o literal %

As sete últimas combinacións de caracteres desta lista tamén poden ser utilizados coa función TIME_FORMAT.

■ TIME_FORMAT

Aplica un formato a unha expresión tipo hora. Funciona igual que DATE_FORMAT, pero só pode levar as combinacións especiais de formato para horas, minutos, segundos e microsegundos. Sintaxe:

TIME_FORMAT(hora, "cadea_de_formato")

■ DAY ou DAYOFMONTH

Devolve o día do mes da data que se pasa como parámetro, en número. Sintaxe:

DAY(data)

DAYOFMONTH(data)

■ DAYOFWEEK

Devolve o número de día da semana da data que se pasa como parámetro. Mostra valores numéricos entre o 1 (para o domingo) e o 7 (para o sábado). Sintaxe:

DAYOFWEEK(data)

■ MONTH

Devolve o mes da data especificada. Sintaxe:

MONTH(data)

■ YEAR

Devolve o ano da data especificada. Sintaxe:

YEAR(data)

Exemplos considerando que agora son as 11:45:00 do 13/12/2015:

```
select date_format(curdate(), '%d/%m/%Y'), time_format(curtime(), '%r');
```

Result Grid		Filter Rows:	Export:
	date_format(curdate(), '%d/%m/%Y')	time_format(curtime(), '%r')	
▶	13/12/2015	11:45:00 AM	

```
select date_format(curdate(), 'Lugo a, %d de %m de %Y'), year(now()), month(now());
```

date_format(curdate(), 'Lugo a, %d de %m de %Y')	year(now())	month(now())
Lugo a, 13 de 12 de 2015	2015	12

1.1.1.2 Funcións de cadeas de caracteres

- **ASCII**

Devolve o código ASCII do primeiro carácter dunha cadea. Sintaxe:

ASCII(cadea)

- **CHAR**

Devolve os caracteres que se corresponden con cada número, segundo a táboa de códigos ASCII. Sintaxe:

CHAR(número1 [, número2] ...)

- **HEX**

Devolve a representación en hexadecimal dunha cadea ou dun valor numérico. Cando o parámetro é unha cadea de caracteres, devolve unha combinación de dous díxitos hexadecimais para cada carácter da cadea. Cando o parámetro é un número, devolve a representación do número en hexadecimal. Sintaxe:

HEX(cadea_caracteres)

HEX(número)

- **CONCAT**

Concatena unha serie de cadeas e devolve unha cadea, ou o valor NULL se algún argumento é NULL. Sintaxe:

CONCAT(cadea1, cadea2 [,cadea3] ...)

- **POSITION**

Busca unha subcadea dentro dunha cadea e devolve a posición na que se atopa a primeira aparición. Sintaxe:

POSITION(subcadea IN cadea)

- **LOCATE**

Busca unha subcadea dentro dunha cadea e devolve a posición na que se atopa a primeira aparición. Pódese pasar un terceiro parámetro indicando a posición a partir da cal se fai a busca. Sintaxe:

LOCATE(subcadea, cadea [, posición])

- **LEFT**

Devolve os n primeiros caracteres da cadea, empezando a ler pola esquerda. Sintaxe:

LEFT(cadea, n)

- **RIGHT**

Devolve os n primeiros caracteres da cadea, empezando a ler pola dereita. Sintaxe:

RIGHT(cadea, n)

- **LENGTH**

Devolve a lonxitude da cadea en caracteres. Sintaxe:

LENGTH(cadea)

- **LOWER**

Pasa a minúsculas todos os caracteres da cadea. Sintaxe:

`LOWER (cadea)`

- **UPPER**

Pasa a maiúsculas todos os caracteres da cadea. Sintaxe:

`UPPER (cadea)`

- **LTRIM**

Elimina os espazos situados á esquerda na cadea. Sintaxe:

`LTRIM (cadea)`

- **RTRIM**

Elimina os espazos situados á dereita na cadea. Sintaxe:

`RTRIM (cadea)`

- **TRIM**

Elimina os espazos situados á esquerda e á dereita na cadea. Sintaxe:

`TRIM (cadea)`

- **REPEAT**

Repite unha cadea n veces. Sintaxe:

`REPEAT (cadea, n)`

- **REPLACE**

Substitúe unha subcadea por outra dentro dunha cadea. Sintaxe:

`REPLACE (cadea, subcadea_inicial, subcadea_final)`

- **SPACE**

Devolve unha cadea composta de n espazos en branco. Sintaxe:

`SPACE (n)`

- **SUBSTRING**

Devolve unha subcadea de lonxitude especificada, empezando dende a posición elixida. Se non se indica lonxitude, se extrae dende a posición indicada ata o final. Sintaxe:

`SUBSTRING (cadea, posición, [lonxitude])`

Exemplos considerando que hoxe é 13/12/2015:

```
select length(trim(' sen espazos ')), length(' con espazos '),  
length(curdate());
```

Result Grid				Filter Rows:	Export:	Wrap Cell Content:
	length(trim(' sen espazos '))	length(' con espazos ')	length(curdate())			
▶	11	15	10			

```
select right(curdate(),2), left(curdate(),4), substring(curdate(),6,2);
```

Result Grid				Filter Rows:	Export:
	right(curdate(),2)	left(curdate(),4)	substring(curdate(),6,2)		
▶	13	2015	12		

```
select upper('Federico'), lower('PASO A MINÚSCULAS');
```

Result Grid				Filter Rows:	Export:
	upper('Federico')	lower('PASO A MINÚSCULAS')			
▶	FEDERICO	paso a minúsculas			

```
select concat('bos' , ' días ', @nome:='Manuel'), substring('www.google.es',5,6);
```

Result Grid	Filter Rows:	Export:	Wrap Cell Co
concat('bos', ' días ', @nome:='Manuel')	substring('www.google.es',5,6)		
bos días Manuel	google		

```
select position('r' in 'Federico'), locate('e','Federico',3);
```

Result Grid	Filter Rows:	Exp
position('r' in 'Federico')	locate('e','Federico',3)	
5	4	

```
select ascii('a'),char(97),hex(ascii('a')),hex(97),hex(255), hex('abz');
```

Result Grid		Filter Rows: <input type="text"/>		Export: Wrap		
	ascii('a')	char(97)	hex(ascii('a'))	hex(97)	hex(255)	hex('abz')
	97	a	61	61	FF	61627A

1.1.1.3 Funcións numéricas

- **ABS**

Obtén o valor absoluto dun número. Cando se pasa como parámetro un valor que non é numérico devolve o valor 0. Sintaxe:

ABS(número)

- **SIGN**

Devolve o valor -1 se o número que se pasa como parámetro é negativo, 1 se é positivo e 0 se o parámetro non é un número, ou é o número 0. Sintaxe:

SIGN(número)

- **SQRT**

Devolve a raíz cadrada do número. Sintaxe:

SQRT(número)

- **POWER**

Calcula potencias de número. Devolve o resultado de elevar o *número1* á potencia de *número2*. Sintaxe:

POWER(número1, número2)

- **MOD**

Devolve o resto da división enteira de dous números que se pasan como parámetros. Sintaxe:

MOD(número1, número2)

Outros operadores para obter o resto da división enteira: *número1*%*número2*.

- **ROUND**

Redondea un número decimal ao enteiro máis próximo. Pódese utilizar un segundo argumento para indicar o número de decimais cos que debe facer o redondeo. Cando *decimais* ten un valor negativo, converte en ceros ese número de díxitos contando dende o punto decimal á esquerda e fai o redondeo no seguinte dígito á esquerda. Sintaxe:

ROUND(número [,decimais])

- **CEILING ou CEIL**

Devolve o menor valor enteiro, non menor có número que se pasa como parámetro.

Sintaxe:

CEIL(número)

CEILING(número)

■ TRUNCATE

Retorna o número truncado co número de decimais que se pasan como parámetro. Cando *decimais* ten un valor negativo, converte en zeros ese número de díxitos contando dende o punto decimal á esquerda. Sintaxe:

TRUNCATE(número, decimais)

■ FLOOR

Devolve o número enteiro máis grande que sexa maior có número que se pasa como parámetro. Sintaxe:

FLOOR(número)

■ RAND

devolve un valor de coma flotante aleatorio no rango $0 \leq \text{valor} \leq 1.0$. Sintaxe:

RAND()

RAND(número)

Cando se pasa un número como parámetro produce unha secuencia repetible sempre que se pase ese mesmo número como parámetro.

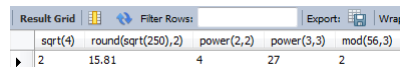
Exemplos:

```
select abs(-90), abs(10), abs('texto'), abs(0), sign(0), sign(-5), sign(8), sign('texto');
```



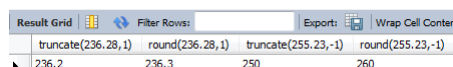
abs(-90)	abs(10)	abs('texto')	abs(0)	sign(0)	sign(-5)	sign(8)	sign('texto')
90	10	0	0	0	-1	1	0

```
select sqrt(4), round(sqrt(250),2), power(2,2), power(3,3), mod(56,3);
```



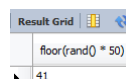
sqrt(4)	round(sqrt(250),2)	power(2,2)	power(3,3)	mod(56,3)
2	15.81	4	27	2

```
select truncate(236.28,1), round(236.28,1), truncate(255.23,-1), round(255.23,-1);
```



truncate(236.28,1)	round(236.28,1)	truncate(255.23,-1)	round(255.23,-1)
236.2	236.3	250	260

```
select floor(rand() * 50); # xera un número de 0 a 49.
```



floor(rand() * 50)
41

```
select dni, rand(5) from empregado;
```

Primeira execución			Segunda execución, e posteriores		
Result Grid			Result Grid		
dni	rand(5)	rand(5)	dni	rand(5)	
33254916	0.40613597483014313	→	33254916	0.40613597483014313	
12852654	0.8745439358749836		12852654	0.8745439358749836	
33123456	0.15431178561813363		33123456	0.15431178561813363	
33456852	0.1479271511993624		33456852	0.1479271511993624	
15245258	0.276700429876056		15245258	0.276700429876056	
33251256	0.9397207265158377		33251256	0.9397207265158377	
33219853	0.8685023736846655		33219853	0.8685023736846655	
33365846	0.5233497196094596		33365846	0.5233497196094596	

```
select dni,rand() from empregado;
```

Primeira execución			Segunda execución		
Result Grid			Result Grid		
dni	rand()	rand()	dni	rand()	
33254916	0.42295907477192496	→	33254916	0.19788184873562478	
12852654	0.8812412022438284		12852654	0.029142507379075984	
33123456	0.13732881763701216		33123456	0.5520670214221506	
33456852	0.042920512187220636		33456852	0.6729076157071698	
15245258	0.8026161387587116		15245258	0.7083370450030426	
33251256	0.8843191879655413		33251256	0.5229624086273484	
33219853	0.013747554285216661		33219853	0.4898009388612592	
33365846	0.41578023826310434		33365846	0.8801174991578958	

1.1.1.4 Funcións de agrupamento ou de columna

Permiten facer cálculos coas columnas da táboa de resultados dunha consulta. Cando se utilizan na lista de selección, non poden ir acompañadas de ningunha outra expresión que non conteña unha función deste tipo. Só poden ir acompañadas de nomes de columnas ou de expresións que conteñan nomes de columnas cando se agrupan filas utilizando a cláusula GROUP BY, que se verá nunha actividade posterior.

■ COUNT

Conta o número de liñas resultantes da consulta, ou o número de valores distintos que toma unha expresión (normalmente, unha columna). Sintaxe:

```
COUNT(*)
```

```
COUNT([DISTINCT] expresión)
```

- Cando se utiliza o símbolo asterisco (*) como parámetro, devolve o número de filas que ten a táboa de resultados.
- Cando se pon como parámetro 'DISTINCT expresión', devolve o número de valores distintos que toma a expresión nas filas da táboa de resultados, sen ter en conta as que toman o valor NULL.
- Cando se pon como parámetro só unha 'expresión', devolve o número de filas da táboa de resultados en que a expresión toma un valor distinto de NULL.

Exemplos:

```
select distinct departamento from empregado;
```


departamento
NULL
1
2
3
4
5
6
7
8
9
10

```
select count(*), count(departamento), count(distinct departamento) from empregado;
```

count(*)	count(departamento)	count(distinct departamento)
20	19	10

O resultados destas consultas informan que hai 20 empregados, 19 empregados cun valor distinto de NULL na columna departamento (hai un empregado que non ten departamento asignado) e hai empregados en 10 departamentos distintos numerados do 1 ao 10.

■ SUM

Suma os valores que toma a expresión (normalmente, unha columna) especificada, para todas a filas resultantes da consulta. A opción DISTINCT non ten en conta os valores repetidos da columna. Sintaxe:

```
SUM ([DISTINCT] expresión)
```

Exemplo: Calcular o que se gasta en salarios. Suma dos salarios brutos.

```
select sum(salario_bruto) from empregado;
```

sum(salario_bruto)
921370.00

■ AVG

Calcula a media dos valores que toma a expresión nas filas da táboa de resultados. A opción DISTINCT non ten en conta os valores repetidos da columna. Sintaxe:

```
AVG ([DISTINCT] expresión)
```

Exemplo: Calcular a media dos salarios brutos dos empregados.

```
select avg(salario_bruto), round(avg(salario_bruto),2) as media_salario from empregado;
```

avg(salario_bruto)	media_salario
46068.500000	46068.50

■ MAX

Devolve o valor máis alto que toma a expresión nas filas da táboa de resultados. Sintaxe:

```
MAX (expresión)
```

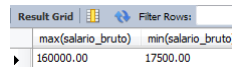
■ MIN

Devolve o valor máis baixo da expresión nas filas da táboa de resultados. Sintaxe:

```
MIN (expresión)
```

Exemplo: Seleccionar o salario máis alto e o máis baixo de todos os empregados.

```
select max(salario_bruto), min(salario_bruto) from empregado;
```



max(salario_bruto)	min(salario_bruto)
160000.00	17500.00

Valores NULL e as funciones de agrupamento:

O estándar ANSI/ISO establece unhas regras para o manexo de valores NULL nas funcións de columna:

- Se algún dos valores dunha columna é NULL, non se ten en conta ao facer os cálculos nunha función de agrupamento.
- Se o valor de todas as columnas é NULL, as funcións SUM, AVG, MAX e MIN devolven o valor NULL, e a función COUNT o valor cero.
- COUNT(*) conta filas e non depende da presenza de valores NULL nas columnas. Cando se quere ter en conta os valores NULL dunha columna, pódese utilizar a fórmula COUNT(expresión) que conta as filas nas que expresión non é NULL.

1.1.1.5 Outras funcións

A continuación móstranse de forma moi resumida grupos de funcións que MySQL incorpora que son utilizados con menos frecuencia pero que poden ser de utilidade.

Funcións de control de fluxo

■ IF

Examina a *expresión1*; e se é verdadeira (*expresión1* <> 0 e *expresión1* <> NULL) entón retorna *expresión2*; se é falsa, retorna *expresión3*. É posible aniñar condicións, de modo que *expresión2* e *expresión3* poden conter funcións IF. Sintaxe:

```
IF(expresión1, expresión2, expresión3)
```

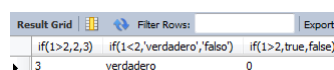
■ IFNULL

Cando a *expresión1* toma o valor NULL, a función devolve o contido de *expresión2*, noutro caso devolve o contido de *expresión1*. Sintaxe:

```
IFNULL(expresión1, expresión2)
```

Exemplos:

```
select if(1>2,2,3), if(1<2,'verdadero','falso'),if(1>2,true,false);
```



if(1>2,2,3)	if(1<2,'verdadero','falso')	if(1>2,true,false)
3	verdadero	0

```
select apellidos, nome, ifnull(departamento,'Sen departamento')  
from empleado  
order by departamento;
```

Result Grid	Filter Rows:	Export:	Wrap C
apellidos	nome	ifnull(departamento,'Sen departamento')	
Iglesias Dominguez	Adolfo	Sen departamento	
Fernandez Diaz	Julian	1	
Martinez Iglesias	Benito	1	
Fernandez Lopez	Jose Luis	1	
Nuñez Bernardéz	Antonia	2	
Martinez Diaz	Carlos	2	

33 12:13:37 select apellidos, nome, ifnull(departamento,'Sen departament... 20 row(s) returned

0.000 sec / 0.000 sec

Funcións de información do sistema

■ USER

Mostra información do usuario que fixo a conexión (nome e host). Sintaxe:

`USER ()`

■ VERSION

Mostra información da versión do servidor MySQL. Sintaxe:

`VERSION ()`

Exemplo:

`select user(), version();`

Result Grid	Filter Rows:
user()	version()
root@localhost	5.6.17

Funcións de cifrado

As funcións de cifrado ou funcións *'hash'* utilízanse para enmascarar información que se desexa ocultar. Unha función *'hash'* é un algoritmo que transforma un conxunto de datos, como pode ser un ficheiro de texto ou unha cadea de caracteres, nun único valor de lonxitude fixa (*'hash'*). O valor *'hash'* calculado pode ser utilizado para verificar a integridade de copias dun dato orixinal sen necesidade de facilitar o dato orixinal. O proceso de cifrado é practicamente irreversible, polo que un valor *'hash'* pode ser libremente distribuído ou almacenado e só se utiliza para fins de comparación. Traballando con bases de datos, o uso máis corrente é para gardar contrasinais.

■ PASSWORD

É a función que usa MySQL para cifrar as contrasinais dos usuarios que acceden ao servidor e que se almacenan na táboa *user* da base de datos *mysql*. Devolve unha cadea de 41 díxitos en hexadecimal. Sintaxe:

`PASSWORD (cadea)`

■ MD5

Calcula unha suma de verificación (checksum) MD5 de 128-bit para unha cadea. Devolve unha cadea de 32 díxitos en hexadecimal. Sintaxe:

`MD5 (cadea)`

■ SHA1 (Secure Hash Algorithm)

Calcula unha suma de verificación (checksum) SHA1 de 160-bit para a cadea. O valor se devolve como unha cadea de 40 díxitos en hexadecimal, ou NULL, no caso de que o parámetro que se pasa tivera o valor NULL. Pode considerarse un equivalente a MD5(), criptograficamente máis seguro. Sintaxe:

SHA1 (cadea)

■ SHA2

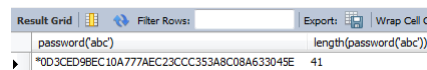
É un conxunto de funcións *'hash'* criptográficas (SHA-224, SHA-256, SHA-384, SHA-512) deseñadas pola 'Agencia de Seguridad Nacional' (NSA). SHA-2 inclúe un importante número de cambios respecto a súa predecesora, SHA-1, que a fai máis segura. Devolve unha cadea en hexadecimal de lonxitude 56, 64, 96, ou 128 caracteres, dependendo da función utilizada. Sintaxe:

SHA2(cadea, lonxitude_resultado)

O primeiro parámetro que se pasa é unha cadea de texto plano, e o segundo indica o tipo de función que se vai a aplicar, e pode tomar os valores 224, 256, 384, 512, ou 0 (que é equivalente a 256). Se algún dos parámetros que se lle pasa toma o valor NULL ou o valor que se pasa como segundo parámetro non é un dos valores permitidos, devolve o valor NULL.

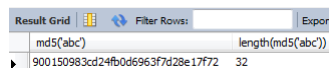
Exemplos:

```
select password('abc'), length(password('abc'));
```



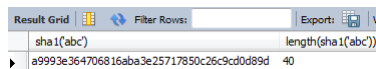
password('abc')	length(password('abc'))
*0D3CED9BEC10A777AECC3CCC353A8C08A633045E	41

```
select md5('abc'), length(md5('abc'));
```



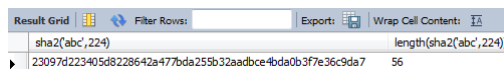
md5('abc')	length(md5('abc'))
900150983cd24fb0d6963f7d28e17f72	32

```
select sha1('abc'), length(sha1('abc'));
```



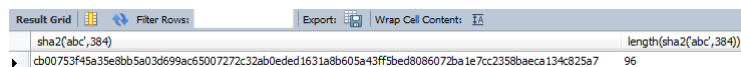
sha1('abc')	length(sha1('abc'))
a9993e364706816aba3e25717850c26c9cd0d89d	40

```
select sha2('abc',224),length(sha2('abc',224));
```



sha2('abc',224)	length(sha2('abc',224))
23097d223405d8228642a477bda255b32aadbce4bda0b3f7e36c9da7	56

```
select sha2('abc',384),length(sha2('abc',384));
```



sha2('abc',384)	length(sha2('abc',384))
cb00753f45a35e8bb5a03d699ac65007272c32ab0eded1631a8b605a43ff5bed8086072ba1e7cc2358baeca134c825a7	96

Funcións de conversión de tipos

Unha expresión ou unha comparación, poden ter operandos de distintos tipos que terán que converterse en tipos compatibles para que se resolva. Algunhas conversións fanse de forma implícita, como por exemplo, a conversión automática de números en cadeas, e viceversa, que fai MySQL.

Exemplo:

```
select 1+'1', concat('proba ', 2);
```

Result Grid		Filter Row
1+1	concat('proba', 2)	
2	proba 2	

As funcións CAST e CONVERT permiten facer conversións explícitas dun tipo de dato a outro, indicando o tipo de dato no que se quere converter. Con estas funcións soluciónase de forma fácil un problema que pode chegar a ser un quebracabezas, sobre todo nas comparacións. Os tipos de datos que se poden utilizar para a conversión son:

- BINARY[(tamaño)]
- CHAR[(tamaño)]
- DATE
- DATETIME
- DECIMAL[(tamaño[,decimais])]
- SIGNED [INTEGER]
- TIME
- UNSIGNED [INTEGER]

As funcións CAST e CONVERT que incorpora MySQL, utilizan a sintaxe SQL estándar.

■ CAST

Devolve o valor que se pasa como primeiro parámetro convertido ao tipo que se indica na función. Sintaxe:

CAST(expresión AS tipo_dato)

As funcións de conversión son útiles, por exemplo, para ordenar os resultados por columnas de tipo ENUM por orden alfabético. Normalmente cando se ordena por unha columna ENUM, utilízanse os valores numéricos do orden interno. Facendo unha conversión a tipo CHAR pódese facer a ordenación por orden alfabético.

Exemplo: seleccionar *codigo*, *nome* e *tipo* de todos os departamentos, ordenando o resultado alfabeticamente pola columna *tipo*, que é de tipo ENUM como se pode ver na descrición do esquema da táboa.

Column	Type
◇ codigo	tinyint(3) unsigned
◇ nome	char(25)
◇ tipo	enum('H','B','A')
◇ cidade	varchar(35)
◇ id_provincia	smallint(6)

A primeira solución ordena polo tipo seguindo o número de orde dos valores válidos (H, B, A).

```
select codigo, nome, tipo
from departamento
order by tipo;
```

Result Grid			Filter R
codigo	nome	tipo	
1	Central	H	
2	Oficina1	H	
8	Oficina7	H	
4	Oficina3	H	
7	Oficina6	H	
10	Oficina9	B	
3	Oficina2	B	
5	Oficina4	A	
9	Oficina8	A	
6	Oficina5	A	
* NULL	NULL	NULL	

A segunda solución ordena alfabeticamente polo valor do tipo transformado en carácter (A, B, H).

```
select codigo, nome, tipo
from departamento
order by cast(tipo as char);
```

Result Grid			Filter I
codigo	nome	tipo	
6	Oficina5	A	
9	Oficina8	A	
5	Oficina4	A	
10	Oficina9	B	
3	Oficina2	B	
4	Oficina3	H	
7	Oficina6	H	
8	Oficina7	H	
2	Oficina1	H	
1	Central	H	
* NULL	NULL	NULL	

■ CONVERT

Transforma unha expresión nun tipo de dato indicado como parámetro. Sintaxe:

CONVERT(expresión, tipo)

CONVERT(expresión USING nome_xogo_carácteres)

- Cando se utiliza a primeira forma da sintaxe, funciona igual que a función CAST, converte a expresión ao tipo de datos que se pasa como segundo parámetro.
- A segunda forma da sintaxe, con USING, utilízase para converter datos entre diferentes xogos de caracteres.

Exemplo: Converter a cadea 'abc' ao xogo de caracteres utf8:

```
select convert('abc' using utf8);
```