Profesora: Olga Cuervo Miguélez

TEMA 4

LENGUAJE SQL. FUNCIONES

Módulo: **Bases de Datos**

Ciclo: **DAM**

11. Funciones de Cadena

En este tema estudiaremos algunas de las **funciones de cadena** de las que dispone MySQL.

Tutorial: https://dev.mysql.com/doc/refman/8.0/en/string-functions.html

1. **ord (carácter):** Retorna el código ASCII para el carácter enviado como argumento. **Ejemplo**:

   select ord ('A'); retorna 65.

2. **char(x,..):** retorna una cadena con los caracteres en código ASCII de los enteros enviados como argumentos. **Ejemplo**:

   select char (65, 66,67); retorna "ABC".

3. **concat (cadena1, cadena2,...):** devuelve la cadena resultado de concatenar los argumentos. **Ejemplo**:

   select concat ('Hola,',' ','como esta?'); retorna "Hola, como esta?".

4. **concat_ws (separador,cadena1,cadena2**,...): "ws" son las iniciales de "with separator". El primer argumento especifica el separador que utiliza para los demás argumentos; el separador se agrega entre las cadenas a concatenar. **Ejemplo**:

   select concat_ws('-','Juan','Pedro','Luis'); retorna "Juan-Pedro-Luis".

5. **find_in_set(cadena,lista de cadenas):** devuelve un valor entre de 0 a n (correspondiente a la posición), si la cadena envidada como primer argumento está presente en la lista de cadenas enviadas como segundo argumento. La lista de cadenas enviada como segundo argumento es una cadena formada por subcadenas separadas por comas. Devuelve 0 si no encuentra "cadena" en la "lista de cadenas". **Ejemplo**:

   select find_in_set('hola','como esta,hola,buen dia'); retorna 2, porque la cadena "hola" se encuentra en la lista de cadenas, en la posición 2.

6. **length(cadena):** retorna la longitud de la cadena enviada como argumento. **Ejemplo**:

   select length('Hola'); devuelve 4.

7. **locate(subcadena,cadena):** retorna la posición de la primera ocurrencia de la subcadena en la cadena enviadas como argumentos. Devuelve "0" si la subcadena no se encuentra en la cadena. **Ejemplo**:

   select locale('o','como le va'); retorna 2.

8. **position(subcadena in cadena):** funciona como "locate()". Devuelve "0" si la subcadena no se encuentra en la cadena. **Ejemplo**:

   select position('o' in 'como le va'); retorna 2.

9. **locate(subcadena,cadena,posicioninicial):** retorna la posición de la primera ocurrencia de la subcadena enviada como primer argumentos en la cadena enviada como segundo argumento, empezando en la posición enviada como tercer argumento. Devuelve "0" si la subcadena no se encuentra en la cadena. **Ejemplos**:

   select locate('ar','Margarita',1); retorna 1.
   select locate('ar','Margarita',3); retorna 5.

10. **instr(cadena,subcadena):** retorna la posición de la primera ocurrencia de la subcadena enviada como segundo argumento en la cadena enviada como primer argumento. **Ejemplo**:

    select instr('como le va','om'); devuelve 2.

11. **lpad(cadena,longitud,cadenarelleno):** retorna la cadena enviada como primer argumento, rellenando por la izquierda con la cadena enviada como tercer argumento hasta que la cadena retornada tenga la longitud especificada como segundo argumento. Si la cadena es más larga, la corta. **Ejemplo**:

    select lpad('hola',10,'0'); retorna "000000hola".

12. **rpad(cadena,longitud,cadenarelleno):** igual que "lpad" excepto que rellena por la derecha.

13. **left(cadena,longitud):** retorna la cantidad (longitud) de caracteres de la cadena comenzando desde la izquierda, primer carácter. **Ejemplo**:

    select left('buenos dias',8); retorna "buenos d".

14. **right(cadena,longitud):** retorna la cantidad (longitud) de caracteres de la cadena comenzando desde la derecha, último carácter. **Ejemplo**:

    select right('buenos dias',8); retorna "nos dias".

15. **substring(cadena,posicion,longitud**): retorna una subcadena de tantos caracteres de longitud como especifica en tercer argumento, de la cadena enviada como primer argumento, empezando desde la posición especificada en el segundo argumento. **Ejemplo**:

    select substring('Buenas tardes',3,5); retorna "enas ".

16. **substring(cadena from posicion for longitud):** variante de "substring(cadena,posicion,longitud)". **Ejemplo**:

    select substring('Buenas tardes' from 3 for 5); retorna "enas".

17. **mid(cadena,posicion,longitud):** igual que "substring(cadena,posicion,longitud)". **Ejemplo**:

    select mid('Buenas tardes' from 3 for 5); retorna "enas t".

18. **substring(cadena,posicion):** retorna la subcadena de la cadena enviada como argumento, empezando desde la posición indicada por el segundo argumento. **Ejemplo**:

    select substring('Margarita',4); retorna "garita".

19. **substring(cadena from posicion):** variante de "substring(cadena,posicion)". **Ejemplo**:

    select substring('Margarita' from 4); retorna "garita".

20. **substring_index(cadena,delimitador,ocurrencia):** retorna la subcadena de la cadena enviada como argumento antes o después de la "ocurrencia" de la cadena enviada como delimitador. Si "ocurrencia" es positiva, retorna la subcadena anterior al delimitador (comienza desde la izquierda); si "ocurrencia" es negativa, retorna la subcadena posterior al delimitador (comienza desde la derecha). **Ejemplo**:

select substring_index('margarita','ar',2); retorna "marg", todo lo anterior a la segunda ocurrencia de "ar".

select substring_index('margarita','ar',-2); retorna "garita", todo lo posterior a la segunda ocurrencia de "ar".

21. **ltrim(cadena):** retorna la cadena con los espacios de la izquierda eliminados. **Ejemplo**:

    select ltrim(' Hola '); retorna "Hola "

22. **rtrim(cadena):** retorna la cadena con los espacios de la derecha eliminados. **Ejemplo**:

    select rtrim(' Hola '); retorna " Hola"

23. **trim([[both|leading|trailing] [subcadena] from] cadena):** retorna una cadena igual a la enviada pero eliminando la subcadena prefijo y/o sufijo. Si no se indica ningún especificador (both, leading o trailing) se asume "both" (ambos). Si no se especifica prefijos o sufijos elimina los espacios. **Ejemplos**:

    select trim(' Hola '); retorna 'Hola'.
    select trim (leading '0' from '00hola00'); retorna "hola00".
    select trim (trailing '0' from '00hola00'); retorna "00hola".
    select trim (both '0' from '00hola00'); retorna "hola".

select trim ('0' from '00hola00'); *retorna "hola".*
select trim (' hola '); *retorna "hola".*

24. **replace(cadena,cadenareemplazo,cadenareemplazar**): retorna la cadena con todas las ocurrencias de la subcadena reemplazo por la subcadena a reemplazar. **Ejemplo**:

    select replace('xxx.mysql.com','x','w'); *retorna "www.mysql.com'.*

25. **repeat(cadena,cantidad):** devuelve una cadena consistente en la cadena repetida la cantidad de veces especificada. Si "cantidad" es menor o igual a cero, retorna una cadena vacía. **Ejemplo**:

    select repeat('hola',3); *retorna "holaholahola".*

26. **reverse(cadena):** devuelve la cadena invirtiendo el order de los caracteres. Ejemplo:

    select reverse('Hola'); *retorna "aloH".*

27. **insert(cadena,posicion,longitud,nuevacadena):** retorna la cadena con la nueva cadena colocándola en la posición indicada por "posicion" y elimina la cantidad de caracteres indicados por "longitud". **Ejemplo**:

    select insert('buenas tardes',2,6,'xx'); *retorna ""bxxtardes".*

28. **lcase(cadena) y lower(cadena):** retornan la cadena con todos los caracteres en minúsculas. **Ejemplo**:

    select lower('HOLA ESTUDIAnte'); *retorna "hola estudiante".*
    select lcase('HOLA ESTUDIAnte'); *retorna "hola estudiante".*

29. **ucase(cadena) y upper(cadena):** retornan la cadena con todos los caracteres en mayúsculas. **Ejemplo**:

    select upper('HOLA ESTUDIAnte'); *retorna "HOLA ESTUDIANTE".*
    select ucase('HOLA ESTUDIAnte'); *retorna "HOLA ESTUDIANTE".*

30. **strcmp(cadena1,cadena2):** retorna 0 si las cadenas son iguales, -1 si la primera es menor que la segunda y 1 si la primera es mayor que la segunda. **Ejemplo**:

    select strcmp('Hola','Chau'); *retorna 1.*

12. Funciones de Fecha-Hora

En este tema estudiaremos algunas de las **funciones de fecha-hora** de las que dispone MySQL.

Tutorial: https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html

- Las funciones que retornan la fecha u hora actuales se evalúan sólo una vez por consulta al principio de la ejecución de consulta. Esto significa que las referencias múltiples a una función tales como **NOW()** en una misma consulta siempre producen el mismo resultado.
- Las funciones **CURRENT_TIMESTAMP()**, **CURRENT_TIME()**, **CURRENT_DATE()**, y **FROM_UNIXTIME()** retornan valores en la zona horaria de la conexión que está disponible como valor de la variable de sistema **time_zone** .
- Los rangos de retorno en las siguientes funciones se aplican en fechas completas. Si la fecha es un valor "`cero`" o una fecha incompleta como **'2001-11-00'**, las funciones que extraen una parte de la fecha pueden retornar 0.

  SELECT DAYOFMONTH('2001-11-00'); *retorna 0.*

1. **ADDDATE(*date*,INTERVAL *expr type*)**

Estas funciones realizan operaciones aritméticas de fechas. **date** es un valor DATETIME o DATE especificando la fecha de inicio. **expr** es una expresión que especifica el intervalo a añadir o borrar de la fecha de inicio. **expr** es una cadena; puede comenzar con un '-' para intervalos negativos. **type** es una palabra clave que indica cómo debe interpretarse la expresión.

La palabra clave **INTERVAL** y el especificador **type** no son sensibles a mayúsculas.

La siguiente tabla muestra cómo se relacionan los argumentos ***type*** y ***expr***:

Para más información puedes consultar este enlace: MySQL :: MySQL 8.0 Reference Manual :: 9.5 Expressions

unit Value	Expected *expr* Format
MICROSECOND	MICROSECONDS
SECOND	SECONDS
MINUTE	MINUTES
HOUR	HOURS
DAY	DAYS
WEEK	WEEKS
MONTH	MONTHS
QUARTER	QUARTERS
YEAR	YEARS
SECOND_MICROSECOND	'SECONDS.MICROSECONDS'
MINUTE_MICROSECOND	'MINUTES:SECONDS.MICROSECONDS'
MINUTE_SECOND	'MINUTES:SECONDS'
HOUR_MICROSECOND	'HOURS:MINUTES:SECONDS.MICROSECONDS'
HOUR_SECOND	'HOURS:MINUTES:SECONDS'
HOUR_MINUTE	'HOURS:MINUTES'
DAY_MICROSECOND	'DAYS HOURS:MINUTES:SECONDS.MICROSECONDS'
DAY_SECOND	'DAYS HOURS:MINUTES:SECONDS'
DAY_MINUTE	'DAYS HOURS:MINUTES'
DAY_HOUR	'DAYS HOURS'
YEAR_MONTH	'YEARS-MONTHS'

Funciones sinónimas:

- **ADDDATE(*expr*,*days*).**
- **DATE_ADD(*date*, INTERVAL *expr type*)**

SELECT ADDDATE('1998-01-02', INTERVAL 31 DAY); →*'1998-02-02'*

SELECT ADDDATE('1998-01-02', 31); → *'1998-02-02'*

SELECT DATE_ADD('1998-01-02', INTERVAL 31 DAY); → *'1998-02-02'*

2. SUBDATE(*date*,INTERVAL *expr type*)

Resta los años, meses o días especificados en el segundo argumento a la fecha especificada en el primer argumento.

Funciones sinónimas:

- **SUBDATE**(*expr*,*days*).
- **DATE_SUB**(*date*, **INTERVAL** *expr type*)

3. ADDTIME(*expr*,*expr2*)

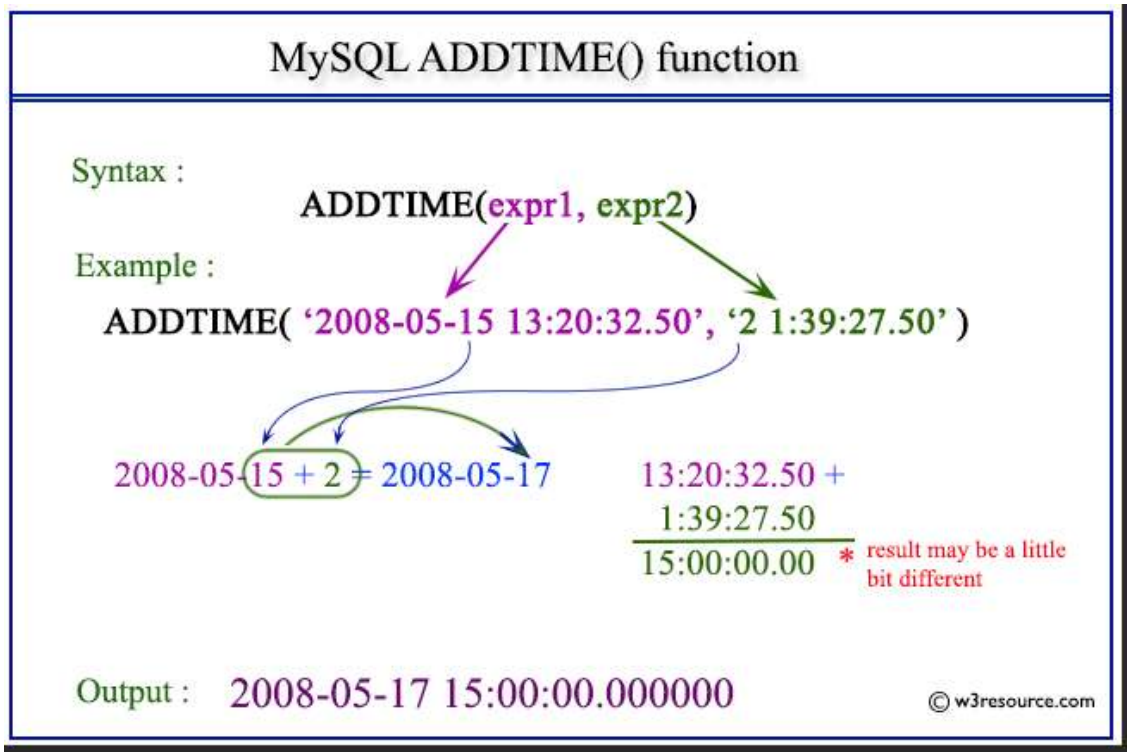
Añade ***expr2*** a ***expr*** y retorna el resultado. ***expr*** es una expresión de tiempo o fecha-hora, y ***expr2*** es una expresión temporal.

SELECT ADDTIME('1997-12-31 23:59:59.999999', '1 1:1:1.000002');

-> '1998-01-02 01:01:01.000001'

SELECT ADDTIME('01:00:00.999999', '02:00:00.999998');

-> '03:00:01.999997'



4. **CURDATE()**

Retorna la fecha horaria como valor en formato **'YYYY-MM-DD'** o **YYYYMMDD**, dependiendo de si la función se usa en un contexto numérico o de cadena de caracteres.

SELECT CURDATE(); *-> '1997-12-15'*

SELECT CURDATE() + 0; *-> 19971215*

Funciones sinónima:

- **CURRENT_DATE**
- **CURRENT_DATE()**

5. **CURTIME()**

Retorna la hora actual como valor en formato **'HH:MM:SS'** o **HHMMSS** dependiendo de si la función se usa en un contexto numérico o de cadena de caracteres.

SELECT CURTIME(); *-> '23:50:26'*

SELECT CURTIME() + 0; *->235026*

Funciones sinónima:

- **CURRENT_TIME**
- **CURRENT_TIME()**

6. **CURRENT_TIMESTAMP**

Retorna un instante en el tiempo.

SELECT CURRENT_TIMESTAMP; *→ 23:03:15 01:26:12*

Funciones sinónima:

- **CURRENT_TIMESTAMP()**
- **NOW()**

7. **DATE(*expr*)**

Extrae la parte de fecha de la expresión de fecha o fecha y hora ***expr***.

SELECT DATE('2003-12-31 01:02:03'); *→'2003-12-31'*

8. DATEDIFF(*expr*,*expr2*)

Retorna el número de días entre la fecha inicial ***expr*** y la fecha final ***expr2***. *expr* y *expr2* son expresiones de fecha o de fecha-hora. Sólo las partes de fecha de los valores se usan en los cálculos.

SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-30');

-> 1

SELECT DATEDIFF('1997-11-30 23:59:59','1997-12-31');

-> -31

9. DATE_FORMAT(*date*,*format*)

Formatea el valor ***date*** según la cadena ***format***. Los siguientes especificadores pueden usarse en la cadena *format*:

Specifier	Description
%a	Abbreviated weekday name (Sun..Sat)
%b	Abbreviated month name (Jan..Dec)
%c	Month, numeric (0..12)
%D	Day of the month with English suffix (0th, 1st, 2nd, 3rd, ...)
%d	Day of the month, numeric (00..31)
%e	Day of the month, numeric (0..31)
%f	Microseconds (000000..999999)
%H	Hour (00..23)
%h	Hour (01..12)
%I	Hour (01..12)
%i	Minutes, numeric (00..59)
%j	Day of year (001..366)
%k	Hour (0..23)
%l	Hour (1..12)
%M	Month name (January..December)
%m	Month, numeric (00..12)
%p	AM or PM
%r	Time, 12-hour (*hh:mm:ss* followed by AM or PM)
%S	Seconds (00..59)
%s	Seconds (00..59)
%T	Time, 24-hour (*hh:mm:ss*)
%U	Week (00..53), where Sunday is the first day of the week; WEEK() mode 0
%u	Week (00..53), where Monday is the first day of the week; WEEK() mode 1
%V	Week (01..53), where Sunday is the first day of the week; WEEK() mode 2; used with %X
%v	Week (01..53), where Monday is the first day of the week; WEEK() mode 3; used with %x
%W	Weekday name (Sunday..Saturday)
%w	Day of the week (0=Sunday..6=Saturday)
%X	Year for the week where Sunday is the first day of the week, numeric, four digits; used with %V
%x	Year for the week, where Monday is the first day of the week, numeric, four digits; used with %v
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)
%%	A literal % character
%*x*	*x*, for any "*x*" not listed above

- Los rangos para los especificadores de mes y día comienzan en cero debido a que MySQL permite almacenar fechas incompletas tales como **'2004-00-00'**.

```
SELECT DATE_FORMAT('1997-10-04 22:23:00', '%W %M %Y');          -> 'Saturday October 1997'
SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H:%i:%s');          -> '22:23:00'
SELECT DATE_FORMAT('1997-10-04 22:23:00', '%D %y %a %d %m %b %j');
  -> '4th 97 Sat 04 10 Oct 277'
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'
SELECT DATE_FORMAT('1999-01-01', '%X %V');    -> '1998 52'
```

10. DAY(*date*)

Retorna el día del mes como número para **date**.

```
SELECT DAYOFMONTH(NOW());        ->27
```

Función sinónimo:

- **DAYOFMONTH(*date*)**.

11. DAYNAME(*date*)

Retorna el nombre del día de la semana para **date**.

```
SELECT DAYNAME('1998-02-05');     -> 'Thursday'
```

12. DAYOFWEEK(*date*)

Retorna el índice del día de la semana para ***date*** (1 = domingo, 2 = lunes, ..., 7 = sábado).

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

13. DAYOFYEAR(*date*)

Retorna el día del año para **date**, en el rango 1 a 366.

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

14. EXTRACT(*type* FROM *date*)

La función **EXTRACT()** usa la misma clase de especificadores de tipo que **DATE_ADD()** o **DATE_SUB()**, pero extrae partes de la fecha en lugar de realizar aritmética de fecha.

```
SELECT EXTRACT(YEAR FROM '1999-07-02');                    -> 1999
SELECT EXTRACT(YEAR_MONTH FROM '1999-07-02 01:02:03');     -> 199907
SELECT EXTRACT(DAY_MINUTE FROM '1999-07-02 01:02:03');     -> 20102
SELECT EXTRACT(MICROSECOND FROM '2003-01-02 10:30:00.000123'); -> 123
```

15. FROM_DAYS(*N*)

Dado un número de día ***N***, retorna un valor **DATE.**

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

Use **FROM_DAYS()** con precaución en fechas viejas. No se pretende que se use con valores que precedan el calendario Gregoriano (1582).

16. GET_FORMAT(DATE|TIME|DATETIME, 'EUR'|'USA'|'JIS'|'ISO'|'INTERNAL')

Retorna una cadena de formato. Esta función es útil en combinación con las funciones **DATE_FORMAT()** y **STR_TO_DATE()**.

Los tres valores posibles para el primer argumento y los cinco posibles valores para el segundo argumento resultan en 15 posibles cadenas de formato (para los especificadores usados, consulte la tabla en la descripción de la función **DATE_FORMAT()**.

La siguiente tabla muestra ejemplos de formatos DATE y TIME en EE. UU., ISO, JIS y EUR. En esta tabla:

- USA se refiere al formato de los Estados Unidos de América.
- ISO se refiere al formato de la Organización Internacional de Normalización.
- JIS se refiere al formato de la Norma Industrial Japonesa.
- EUR se refiere al formato europeo.

Function Call	Result
GET_FORMAT(DATE,'USA')	'%m.%d.%Y'
GET_FORMAT(DATE,'JIS')	'%Y-%m-%d'
GET_FORMAT(DATE,'ISO')	'%Y-%m-%d'
GET_FORMAT(DATE,'EUR')	'%d.%m.%Y'
GET_FORMAT(DATE,'INTERNAL')	'%Y%m%d'
GET_FORMAT(DATETIME,'USA')	'%Y-%m-%d %H.%i.%s'
GET_FORMAT(DATETIME,'JIS')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(DATETIME,'ISO')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(DATETIME,'EUR')	'%Y-%m-%d %H.%i.%s'
GET_FORMAT(DATETIME,'INTERNAL')	'%Y%m%d%H%i%s'
GET_FORMAT(TIME,'USA')	'%h:%i:%s %p'
GET_FORMAT(TIME,'JIS')	'%H:%i:%s'
GET_FORMAT(TIME,'ISO')	'%H:%i:%s'
GET_FORMAT(TIME,'EUR')	'%H.%i.%s'
GET_FORMAT(TIME,'INTERNAL')	'%H%i%s'

TIMESTAMP puede usarse con **GET_FORMAT()** retorna los mismos valores que para **DATETIME**.

```
SELECT DATE_FORMAT('2003-10-03',GET_FORMAT(DATE,'EUR'));        -> '03.10.2003'
SELECT STR_TO_DATE('10.31.2003',GET_FORMAT(DATE,'USA'));        -> '2003-10-31'
```

17. HOUR(*time*)

Retorna la hora para ***time***. El rango del valor de retorno es 0 a 23 para valores de horas del día.

```
SELECT HOUR('10:05:03');  -> 10
```

18. LAST_DAY(*date*)

Toma una fecha o fecha-hora y retorna el valor correspondiente para el último día del mes. Retorna **NULL** si el argumento es inválido.

```
SELECT LAST_DAY('2003-02-05');        -> '2003-02-28'
```

```
SELECT LAST_DAY('2004-02-05');          -> '2004-02-29'
SELECT LAST_DAY('2004-01-01 01:01:01'); -> '2004-01-31'
SELECT LAST_DAY('2003-03-32');          -> NULL
```

19. MAKEDATE(*year*,*dayofyear*)

Retorna una fecha, dado un año y día del año. ***dayofyear*** debe ser mayor a 0 o el resultado es **NULL**.

```
SELECT MAKEDATE(2001,31), MAKEDATE(2001,32);
                -> '2001-01-31', '2001-02-01'
SELECT MAKEDATE(2001,365), MAKEDATE(2004,365);
                -> '2001-12-31', '2004-12-30'
SELECT MAKEDATE(2001,0)          -> NULL
```

20. MAKETIME(*hour*,*minute*,*second*)

Retorna un valor horario calculado a partir de los argumentos ***hour, minute, y second***.

```
SELECT MAKETIME(12,15,30);        -> '12:15:30'
```

21. MINUTE(*time*)

Retorna el minuto de ***time***, en el rango **0** a **59**.

```
SELECT MINUTE('98-02-03 10:05:03');        -> 5
```

22. MONTH(*date*)

Retorna el mes para ***date***, en el rango **1** a **12**.

```
SELECT MONTH('1998-02-03');                →2
```

23. MONTHNAME(*date*)

Retorna el nombre completo del mes para ***date***.

```
SELECT MONTHNAME('1998-02-05'); -> 'February'
```

24. NOW()

Retorna la fecha y hora actual como valor en formato **'YYYY-MM-DD HH:MM:SS'** o **YYYYMMDDHHMMSS** , dependiendo de si la función se usa en contexto numérico o de cadena de caracteres.

SELECT NOW(); *-> '1997-12-15 23:50:26'*

SELECT NOW() + 0; *-> 19971215235026*

25. PERIOD_ADD(*P*,*N*)

Añade ***N*** meses al periodo ***P*** (en el formato **YYMM** o **YYYYMM**). Retorna un valor en el formato **YYYYMM**. Tenga en cuenta que el argumento del periodo ***P*** *no* es una fecha.

SELECT PERIOD_ADD(9801,2); *-> 199803*

26. PERIOD_DIFF(*P1*,*P2*)

Retorna el número de meses entre periodos **P1** y ***P2***. *P1* y *P2* deben estar en el formato **YYMM** o **YYYYMM**. Tenga en cuenta que los argumentos del periodo P1 *y* P2 no *son fechas.*

SELECT PERIOD_DIFF(9802,199703); *-> 11*

27. QUARTER(*date*)

Retorna el cuarto del año para ***date***, en el rango **1** a **4**.

SELECT QUARTER('98-04-01'); *-> 2*

28. SECOND(*time*)

Retorna el segundo para **time**, en el rango **0** a **59**.

SELECT SECOND('10:05:03'); *-> 3*

29. SEC_TO_TIME(*seconds*)

Retorna el argumento ***seconds***, convertido a horas, minutos y segundos, como un valor en formato **'HH:MM:SS'** o **HHMMSS**, dependiendo de si la función se usa en contexto numérico o de cadena de caracteres.

SELECT SEC_TO_TIME(2378); *-> '00:39:38'*

SELECT SEC_TO_TIME(2378) + 0; *->3938*

30. **STR_TO_DATE(*str*,*format*)**

Esta es la inversa de la función **DATE_FORMAT()**. Toma la cadena ***str*** y la cadena de formato ***format***. **STR_TO_DATE()** retorna un valor **DATETIME** si la cadena de formato contiene parte de fecha y hora, o un valor **DATE** o **TIME** si la cadena contiene sólo parte de fecha o hora.

Los valores fecha, hora o fecha/hora contenidos en ***str*** deben ser dados en el formato indicado por ***format***. Para los especificadores que pueden usarse en *format*, consulte la tabla en la descripción de la función **DATE_FORMAT()** . Todos los otros caracteres no se interpretan. Si *str* contiene un valor fecha, hora o fecha/hora ilegal, **STR_TO_DATE()** retorna **NULL**. A partir de MySQL 5.0.3, un valor ilegal también produce una advertencia.

SELECT STR_TO_DATE('03.10.2003 09.20','%d.%m.%Y %H.%i');

-> '2003-10-03 09:20:00'

SELECT STR_TO_DATE('10arp', '%carp');

-> '0000-10-00 00:00:00'

SELECT STR_TO_DATE('2003-15-10 00:00:00','%Y-%m-%d %H:%i:%s'); -> NULL

31. **SUBTIME(*expr*,*expr2*)**

SUBTIME() resta ***expr2*** de ***expr*** y retorna el resultado. ***expr*** es una expresión de hora o fecha/hora, y ***expr2*** es una expresión de hora.

SELECT SUBTIME('1997-12-31 23:59:59.999999','1 1:1:1.000002');

-> '1997-12-30 22:58:58.999997'

SELECT SUBTIME('01:00:00.999999', '02:00:00.999998');

-> '-00:59:59.999999'

32. **TIME(*expr*)**

Extrae la parte de hora de la expresión hora o fecha/hora ***expr***.

SELECT TIME('2003-12-31 01:02:03'); *-> '01:02:03'*

SELECT TIME('2003-12-31 01:02:03.000123'); *-> '01:02:03.000123'*

33. TIMEDIFF(*expr*,*expr2*)

TIMEDIFF() retorna el tiempo entre la hora de inicio ***expr*** y la hora final ***expr2***. *expr* y *expr2* son expresiones de hora o de fecha/hora, pero ambas deben ser del mismo tipo.

```
SELECT TIMEDIFF('2000:01:01 00:00:00', '2000:01:01 00:00:00.000001');        -> '-00:00:00.000001'
SELECT TIMEDIFF('1997-12-31 23:59:59.000001', '1997-12-30 01:01:01.000002'); -> '46:58:57.999999'
```

34. TIMESTAMP(*expr*) , TIMESTAMP(*expr*,*expr2*)

Con un único argumento, esta función retorna la expresión de fecha o fecha/hora *expr* como valor fecha/hora. Con dos argumentos, suma la expresión de hora *expr2* a la expresión de fecha o de fecha/hora *expr* y retorna el resultado como valor fecha/hora.

```
SELECT TIMESTAMP('2003-12-31');                    -> '2003-12-31 00:00:00'
SELECT TIMESTAMP('2003-12-31 12:00:00','12:00:00'); -> '2004-01-01 00:00:00'
```

35. TIMESTAMPADD(*interval*,*int_expr*,*datetime_expr*)

Suma la expresión entera ***int_expr*** a la expresión de fecha o de fecha/hora ***datetime_expr***. ***int_expr*** la da el argumento a **interval**, que debe ser uno de los siguientes valores: **FRAC_SECOND**, **SECOND**, **MINUTE**, **HOUR**, **DAY**, **WEEK**, **MONTH**, **QUARTER**, o **YEAR**.

El valor ***interval*** puede especificarse usando una de las palabras claves que se muestran, o con un prefijo de **SQL_TSI_**. Por ejemplo, **DAY** o **SQL_TSI_DAY** son legales.

```
SELECT TIMESTAMPADD(MINUTE,1,'2003-01-02');  -> '2003-01-02 00:01:00'
SELECT TIMESTAMPADD(WEEK,1,'2003-01-02');    -> '2003-01-09'
```

36. TIMESTAMPDIFF(*interval*,*datetime_expr1*,*datetime_expr2*)

Retorna la diferencia entera entre las expresiones de fecha o de fecha/hora ***datetime_expr1*** y **datetime_expr2**. La unidad del resultado se da en el argumento ***interval***. Los valores legales para **interval** son los mismos que los listados en la descripción de la función **TIMESTAMPADD()**.

```
SELECT TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01');  -> 3
SELECT TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01');   -> -1
```

37. **TIME_FORMAT(*time*,*format*)**

Se usa como la función **DATE_FORMAT()** pero la cadena ***format*** puede contener sólo los especificadores de formato que tratan horas, minutos y segundos. Otros especificadores producen un valor **NULL** o **0**.

Si el valor time *contiene* una parte horaria mayor que **23**, los especificadores de formato horario **%H** y **%k** producen un valor mayor que el rango usual de **0..23**. Los otros especificadores de hora producen la hora modulo 12.

```
SELECT TIME_FORMAT('100:00:00', '%H %k %h %I %l');        -> '100 100 04 04 4'
```

38. **TIME_TO_SEC(*time*)**

Retorna el argumento ***time*** convertido en segundos.

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

39. **TO_DAYS(*date*)**

Dada la fecha ***date***, retorna un número de día (el número de días desde el año 0).

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

TO_DAYS() no está pensado para usarse con valores anteriores al calendario Gregoriano (1582), ya que no tiene en cuenta los días perdidos cuando se cambió el calendario.

Recuerde que MySQL convierte años de dos dígitos en fechas de cuatro dígitos así por ejemplo, **'1997-10-07'** y **'97-10-07'** se consideran fechas idénticas:

```
SELECT TO_DAYS('1997-10-07'), TO_DAYS('97-10-07');        -> 729669, 729669
```

Para fechas anteriores a 1582 (y posiblemente un año posterior en otras localizaciones), los resultados de esta función no son fiables.

40. **WEEK(*date*[,*mode*])**

Esta función retorna el número de semana para ***date***. La forma de dos argumentos de **WEEK()** le permite especificar si la semana comienza en lunes o domingo y si el valor de retorno debe estar en el rango de **0** a **53** o de **1** a **53**.

La siguiente tabla describe cómo funciona el argumento **mode**:

Modo	Primer día de la semana	Alcance	La semana 1 es la primera semana ...
0	Domingo	0-53	con un domingo de este año
1	Lunes	0-53	con 4 o más días este año
2	Domingo	1-53	con un domingo de este año
3	Lunes	1-53	con 4 o más días este año
4	Domingo	0-53	con 4 o más días este año
5	Lunes	0-53	con un lunes en este año
6	Domingo	1-53	con 4 o más días este año
7	Lunes	1-53	con un lunes en este año

```
SELECT WEEK('1998-02-20');          -> 7
SELECT WEEK('1998-02-20',0);        -> 7
SELECT WEEK('1998-02-20',1);        -> 8
SELECT WEEK('1998-12-31',1);        -> 53
```

41. WEEKDAY(*date*)

Retorna el índice de días de la semana para *date* (**0** = lunes, **1** = martes, ... **6** = domingo).

```
SELECT WEEKDAY('1998-02-03 22:23:00');     -> 1
SELECT WEEKDAY('1997-11-05');              -> 2
```

42. WEEKOFYEAR(*date*)

Retorna la semana de la fecha como número del rango **1** a **53**. Esta es una función de compatibilidad equivalente a **WEEK(date,3)**.

```
SELECT WEEKOFYEAR('1998-02-20');->8
```

43. YEAR(*date*)

Retorna el año para *date*, en el rango **1000** a **9999**.

```
SELECT YEAR('98-02-03');  -> 1998
```

44. YEARWEEK(*date*), YEARWEEK(*date*,*start*)

Retorna año y semana para una fecha. El argumento *start* funciona exactamente como el argumento start de **WEEK()**. El año en el resultado puede ser diferente del año en el argumento fecha para la primera y última semana del año.

```
SELECT YEARWEEK('2015-01-01',0);          -- -> 201452
SELECT YEARWEEK('2015-01-01',1);          -- -> 201501
```

¿Cuántos años tienes hoy?

```
SELECT timestampdiff(YEAR,'2016-04-27',CURDATE()) AS EDAD; -- SIN TENER EN CUENTA LA HORA DE NACIMIENTO

SELECT timestampdiff(YEAR,'2016-04-27 22:41:10',NOW()) AS EDAD; -- TENIENDO EN CUENTA LA HORA DE NACIMIENTO
```

13. Funciones Matemáticas

a. Operadores Aritméticos

La precisión del resultado esta determinada por las siguientes reglas:

- Para los operadores -, + y *, el resultado es calculado con precisión BINGINT (64-bits) sí ambos argumentos son enteros.
- Sí uno de los argumentos es un entero sin signo y el otro es un entero, el resultado es un entero sin signo.
- Sí alguno de los operandos de +, -, /, *, % es un valor real o una cadena, entonces la precisión del resultado es la precisión del argumentos con precisión máxima.
- En la división, la precisión de los resultados cuando se utiliza dos valores exactos es la precisión del primer argumento + la precisión del valor de la variable global **div_precision_increment**.

Por ejemplo, la expresión 5.05 / 0.0014 tendría una precisión de seis cifras decimales (3607.142857).

***show variables**; Permite consultar las variables del sistema y sus valores.*

***set div_precision_increment = 6;** Permite modificar el valor de las variables del sistema.*

- **Suma: +**

  SELECT 3+5; *->8*

- **Resta: -**

  SELECT 3-5; *->-2*

- **Operador unario: -**

Cambia el signo del argumento.

SELECT - 2; *-> -2*

- **Multiplicación: ***

*SELECT 3*5;* *-> 15*

*SELECT 18014398509481984*18014398509481984.0;* *-> 324518553658426726783156020576256.0*

*SELECT 18014398509481984*18014398509481984;* *-> ERROR. El resultado de la última expresión es incorrecto ya que el resultado de la multiplicación entera excede el rango de 64-bit de cálculos BIGINT.*

El resultado de la última expresión es incorrecto ya que el resultado de la multiplicación entera excede el rango de 64-bit de cálculos **BIGINT**.

- **División: /**

  SELECT 3/5; *-> 0.60*

  División por cero produce un resultado **NULL**:

  SELECT 102/(1-1); *-> NULL*

- **DIV**

  División entera. Similar **FLOOR ()** pero funciona con valores **BIGINT**.

  SELECT 5 DIV 2; *-> 2*

b. Funciones Matemáticas

Todas las funciones matemáticas retornan **NULL** en caso de error.

- **ABS(*X*)**. Retorna el valor absoluto de ***X***. Esta función puede usar valores **BIGINT**.

  SELECT ABS(2); *-> 2*

  SELECT ABS(-32); *-> 32*

- **ACOS(*X*)**. Retorna el arcocoseno de ***X***, esto es, el valor cuyo coseno es ***X***. Retorna **NULL** *si* X *no* está en el rango -1 a 1.

  SELECT ACOS(1); *-> 0*

  SELECT ACOS(1.0001); *-> NULL*

  SELECT ACOS(0); *-> 1.5707963267949*

- **ASIN(X)**. Retorna el arcoseno de ***X***, esto es, el valor cuyo seno es *X*. Retorna **NULL** si *X* no está en el rango de -1 a 1.

  SELECT ASIN(0.2); *-> 0.20135792079033*

  SELECT ASIN('foo'); *-> 0*

- **ATAN(*X*)**. Retorna la arcotangente de **X***,* esto es, el valor cuya tangente es *X*.

  SELECT ATAN(2); *-> 1.1071487177941*

  SELECT ATAN(-2); *-> -1.1071487177941*

- **ATAN(*Y,X*)** , **ATAN2(Y,X)**. Retorna la arcotangente de las variables ***X*** y ***Y***. Es similar a calcular la arcotangente de *Y* **/** *X*, excepto que los signos de ambos argumentos se usan para determinar el cuadrante del resultado.

  SELECT ATAN(-2,2); *-> -0.78539816339745*

SELECT ATAN2(PI(),0); *-> 1.5707963267949*

- **CEILING(X)**, **CEIL(*X*)**. Retorna el entero más pequeño no menor a ***X***. Estas dos funciones son sinónimas. Tener en cuenta que el valor retornado se convierte a **BIGINT**.

  SELECT CEILING(1.23); *-> 2*

  SELECT CEIL(-1.23); *-> -1*

- **COS(X)**. Retorna el coseno de ***X***, donde *X* se da en radianes.

  SELECT COS(PI()); *-> -1*

- **COT(*X*)**. Retorna la cotangente de ***X*.**

  SELECT COT(12); *-> -1.5726734063977*

  SELECT COT(0); *-> NULL*

- **DEGREES(*X*)**. Retorna el argumento ***X*,** convertido de radianes a grados.

  SELECT DEGREES(PI()); *-> 180*

  SELECT DEGREES(PI() / 2); *-> 90*

- **EXP(X)**. Retorna el valor de **e** (la base del logaritmo natural o neperiano) a la potencia de ***X***.

  SELECT EXP(2); *-> 7.3890560989307*

  SELECT EXP(-2); *-> 0.13533528323661*

- **FLOOR(*X*)**. Retorna el valor entero más grande pero no mayor a ***X***. Tenga en cuenta que el valor devuelto se convierte a **BIGINT**.

  SELECT FLOOR(1.23); *-> 1*

  SELECT FLOOR(-1.23); *-> -2*

- **LN(X).** Esta función es sinónimo a **LOG(*X*)**. Retorna el logaritmo natural de ***X***, esto es, el logaritmo de *X* base *e*.

  SELECT LN(2); *-> 0.69314718055995*

  SELECT LN(-2); *-> NULL*

- **LOG(*X*)**, **LOG(B,X)**. Si se invoca con un parámetro, esta función retorna el logaritmo natural de ***X*.**

  SELECT LOG(2); *-> 0.69314718055995*

  SELECT LOG(-2); *-> NULL*

  Si se llama con dos parámetros, esta función retorna el logaritmo de X para una base B.

  SELECT LOG(2,65536); *-> 16*

  SELECT LOG(10,100); *-> 2*

  LOG(*B,X*) es equivalente a **LOG(*X*) / LOG(*B*)**.

- **LOG2(X)**. Retorna el logaritmo en base 2 de *X*.

  SELECT LOG2(65536); *-> 16*

  SELECT LOG2(-100); *-> NULL*

  LOG2() es útil para encontrar cuántos bits necesita un número para almacenamiento. Esta función es equivalente a la expresión **LOG(*X*) / LOG(2)**.

- **LOG10(*X*)**. Retorna el logaritmo en base 10 de ***X*.**

  SELECT LOG10(2); *-> 0.30102999566398*

  SELECT LOG10(100); *-> 2*

  SELECT LOG10(-100); *-> NULL*

  LOG10(*X*) es equivalente a **LOG(10,*X*)**.

- **MOD(N,M)** , ***N* % M**, ***N* MOD M.** Operación de módulo. Retorna el resto de ***N*** dividido por ***M***. Esta función puede usar valores **BIGINT**.

  SELECT MOD(234, 10); *-> 4*

  SELECT 253 % 7; *-> 1*

  SELECT MOD(29,9); *-> 2*

  SELECT 29 MOD 9; *-> 2*

  MOD() también funciona con valores con una parte fraccional y retorna el resto exacto tras la división:

  SELECT MOD(34.5,3); *-> 1.5*

- **PI()**. Retorna el valor del número (**pi**). El número de decimales que se muestra por defecto es siete, pero MySQL usa internamente el valor de doble precisión entero.

  SELECT PI(); *-> 3.141593*

  SELECT PI()+0.000000000000000000; *-> 3.141592653589793116*

- **POW(X,Y)** , **POWER(*X*,*Y*)**. Retorna el valor de ***X*** a la potencia de ***Y*.**

  SELECT POW(2,2); *-> 4*

  SELECT POW(2,-2); *-> 0.25*

- **RADIANS(*X*)**. Retorna el argumento *X*, convertido de grados a radianes. (Tenga en cuenta que π radianes son 180 grados.)

  SELECT RADIANS(90); *-> 1.5707963267949*

- **RAND()**, **RAND(N)**. Retorna un valor aleatorio en coma flotante del rango de 0 a 1.0.

```
SELECT RAND();                -- -> 0.9233482386203
```

Puede usar esta función para recibir registros de forma aleatoria como se muestra aquí:

```
CREATE TABLE test.t (i INT);
INSERT INTO test.t VALUES(1),(2),(3);
SELECT i, RAND(2) FROM test.t;
```

- ***ROUND(X)***, ***ROUND(X,D)***. Retorna el argumento ***X***, redondeado al entero más cercano. *Con dos argumentos, retorna* **X** redondeado a **D** decimales.

```
SELECT ROUND(-1.23);          -> -1
SELECT ROUND(-1.58);          -> -2
SELECT ROUND(1.58);           -> 2
SELECT ROUND(1.298, 1);       -> 1.3
SELECT ROUND(1.298, 0);       -> 1
```

- **SIGN(*X*)**. Retorna el signo del argumento como **-1**, **0**, o **1**, en función de si ***X*** es negativo, cero o positivo.

```
SELECT SIGN(-32);             -> -1
SELECT SIGN(0);               -> 0
SELECT SIGN(234);             -> 1
```

- **SIN(X)**. Retorna el seno de ***X***, donde *X* se da en radianes.

```
SELECT SIN(PI());             -> 1.2246063538224e-16
SELECT ROUND(SIN(PI()));      -> 0
```

- **SQRT(X).** Retorna la raíz cuadrada de un número no negativo ***X***.

```
SELECT SQRT(4);               -> 2
SELECT SQRT(20);              -> 4.4721359549996
SELECT SQRT(-16);             -> NULL
```

- **TAN(X).** Retorna la tangente de ***X***, donde *X* se da en radianes.

```
SELECT TAN(PI());             -> -1.2246063538224e-16
```

SELECT TAN(PI()+1); *-> 1.5574077246549*

- **TRUNCATE(X,D).** Retorna el número ***X***, truncado a ***D*** decimales. Si *D* es 0, el resultado no tiene parte decimal.

  SELECT TRUNCATE(1.223,1); *-> 1.2*

  SELECT TRUNCATE(1.999,1); *-> 1.9*

  SELECT TRUNCATE(1.999,0); *-> 1*

  SELECT TRUNCATE(-1.999,1); *-> -1.9*

ÍNDICE