Funciones de fecha y hora MySQL

ADDDATE() Suma un intervalo de tiempo a una fecha

ADDTIME() Suma tiempos

CONVERT_TZ() Convierte tiempos entre distintas zonas horarias

CURDATE() o CURRRENT_DATE() Obtener la fecha actual

CURTIME() o CURRENT_TIME() Obtener la hora actual

DATE() Extraer la parte correspondiente a la fecha

DATEDIFF() Calcula la diferencia en días entre dos fechas

DATE_ADD() Aritmética de fechas, suma un intervalo de tiempo

DATE_SUB() Aritmética de fechas, resta un intervalo de tiempo

DATE_FORMAT() Formatea el valor de una fecha

DAY() o DAYOFMONTH() Obtiene el día del mes a partir de una fecha

DAYNAME() Devuelve el nombre del día de la semana

DAYOFWEEK() Devuelve el índice del día de la semana

DAYOFYEAR() Devuelve el día del año para una fecha

EXTRACT() Extrae parte de una fecha

FROM_DAYS() Obtener una fecha a partir de un número de días

FROM_UNIXTIME() Representación de fechas UNIX en formato de cadena

GET_FORMAT() Devuelve una cadena de formato

HOUR() Extrae la hora de un valor time

LAST_DAY() Devuelve la fecha para el último día del mes de una fecha

MAKEDATE() Calcula una fecha a partir de un año y un día del año

MAKETIME() Calcula un valor de tiempo a partir de una hora, minuto y segundo

MICROSECOND() Extrae los microsegundos de una expresión de fecha/hora o de hora

MINUTE() Extrae el valor de minutos de una expresión time

MONTH() Devuelve el mes de una fecha

MONTHNAME() Devuelve el nombre de un mes para una fecha

NOW() o CURRENT_TIMESTAMP() o LOCALTIME() o LOCALTIMESTAMP() o SYSDATE() Devuelve la fecha y hora actual

PERIOD_ADD() Añade meses a un periodo (año/mes)

PERIOD_DIFF() Calcula la diferencia de meses entre dos periodos (año/mes)

QUARTER() Devuelve el cuarto del año para una fecha

SECOND() Extrae el valor de segundos de una expresión time

SEC_TO_TIME() Convierte una cantidad de segundos a horas, minutos y segundos

STR_TO_DATE() Obtiene un valor DATETIME a partir de una cadena con una fecha y una cadena de formato

SUBDATE() Resta un intervalo de tiempo de una fecha

SUBTIME() Resta dos expresiones time

TIME() Extrae la parte de la hora de una expresión fecha/hora

TIMEDIFF() Devuelve en tiempo entre dos expresiones de tiempo

TIMESTAMP() Convierte una expresión de fecha en fecha/hora o suma un tiempo a una fecha

TIMESTAMPADD() Suma un intervalo de tiempo a una expresión de fecha/hora

TIMESTAMPDIFF() Devuelve la diferencia entre dos expresiones de fecha/hora

TIME_FOMAT() Formatea un tiempo

TIME_TO_SEC() Convierte un tiempo a segundos

TO_DAYS() Calcula el número de días desde el año cero

UNIX_TIMESTAMP() Devuelve un timestamp o una fecha en formato UNIX, segundos desde 1070

UTC_DATE() Devuelve la fecha UTC actual

UTC_TIME() Devuelve la hora UTC actual

UTC_TIMESTAMP() Devuelve la fecha y hora UTC actual

WEEK() Calcula el número de semana para una fecha

WEEKDAY() Devuelve el número de día de la semana para una fecha

WEEKOFYEAR() Devuelve el número de la semana del año para una fecha

YEAR() Extrae el año de una fecha

YEARWEEK() Devuelve el año y semana de una fecha

Funciones para presentación de datos de fecha y hora

CONVERT_TZ

Formato: CONVERT_TZ(dt, from_tz,to_tz)

CONVERT_TZ() convierte un valor dt, de fecha y hora, de la zona de tiempos **from_tz** dada a la zona de tiempos **to_tz** y devuelve el valor resultante. Esta función devuelve *NULL* si el argumento es inválido. Si el valor cae fuera del rango soportado por el tipo *TIMESTAMP* cuando se convierte desde from_tz a UTC, no se realiza conversión alguna.

```
SELECT CONVERT_TZ('2004-01-01 12:00:00','GMT','MET');
-> '2004-01-01 13:00:00'
SELECT CONVERT_TZ('2004-01-01 12:00:00','+00:00','-07:00');
-> '2004-01-01 05:00:00'
```

Para usar zonas de tiempo con nombre como 'MET' o 'Europe/Moscow', la tabla de zonas de tiempo debe ser preparada convenientemente.

CURDATE() / CURRENT_DATE / CURRENT_DATE()

CURDATE()

CURRENT_DATE

CURRENT_DATE()

Devuelve la fecha actual como un valor en el formato 'AAAA-MM-DD' o AAAAMMDD, dependiendo de si la función se usa en un contexto de cadena o numérico.

CURRENT_DATE y **CURRENT_DATE()** son sinónimos de **CURDATE**.

```
SELECT CURDATE(), CURDATE() + 0, CURRENT_DATE(), CURRENT_DATE
```

CURTIME() / CURRENT_TIME / CURRENT_TIME()

CURTIME()

CURRENT_TIME

CURRENT_TIME()

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

CURRENT_TIME y CURRENT_TIME() son sinónimos de CURTIME().

```
SELECT CURTIME(), CURTIME() + 0, CURRENT_TIME(), CURRENT_TIME;
```

NOW() / CURRENT_TIMESTAMP / CURRENT_TIMESTAMP() LOCALTIME / LOCALTIME() / LOCALTIMESTAMP LOCALTIMESTAMP() / SYSDATE()

NOW()

CURRENT_TIMESTAMP

CURRENT_TIMESTAMP()

LOCALTIME

LOCALTIME()

LOCALTIMESTAMP

LOCALTIMESTAMP()

SYSDATE()

Devuelve la fecha y hora actual como un valor en el formato 'YYYY-MM-DD HH:MM:SS' o YYYYMMDDHHMMSS, dependiendo de si la función se usa en un contexto de cadena o de número.

CURRENT_TIMESTAMP, **CURRENT_TIMESTAMP()**, **LOCALTIME**, **LOCALTIME()**, **LOCALTIMESTAMP**, **LOCALTIMESTAMP()** y **SYSDATE()** son sinónimos de **NOW()**.

```
SELECT NOW(), NOW() + 0, CURRENT_TIMESTAMP, CURRENT_TIMESTAMP(), LOCALTIME, LOCALTIME(), LOCALTIMESTAMP, LOCALTIMESTAMP(), SYSDATE()
```

DATE()

DATE(expr)

Extrae la parte de la fecha de una expresión expre de tipo date o datetime.

```
SELECT DATE('2011-12-31 01:02:03'), DATE(NOW());
```

DATE_FORMAT()

DATE_FORMAT(fecha, formato)

Formatea el valor de fecha de acuerdo con la cadena de formato. Se pueden usar los siguientes especificadores para la cadena de formato:

Especificador	Descripción
%M	Nombre del mes (January..December)
%W	Nombre de día (Sunday..Saturday)
%D	Día del mes con sufijo en inglés (0th, 1st, 2nd, 3rd, etc.)
%Y	Año, numérico con 4 dígitos
%y	Año, numérico con 2 dígitos

%X	Año para la semana donde el domingo es el primer día de la semana, numérico de 4 dígitos; usado junto con %V
%x	Año para la semana donde el lunes es el primer día de la semana, numérico de 4 dígitos; usado junto con %v
%a	Nombre de día de semana abreviado (Sun..Sat)
%d	Día del mes, numérico (00..31)
%e	Día del mes, numérico (0..31)
%m	Mes, numérico (00..12)
%c	Mes, numérico (0..12)
%b	Nombre del mes abreviado (Jan..Dec)
%j	Día del año (001..366)
%H	Hora (00..23)
%k	Hora (0..23)
%h	Hora (01..12)
%I	Hora (01..12)
%l	Hora (1..12)
%i	Minutos, numérico (00..59)
%r	Tiempo, 12-horas (hh:mm:ss seguido por AM o PM)
%T	Tiempo, 24-horas (hh:mm:ss)
%S	Segundos (00..59)
%s	Segundos (00..59)
%f	Microsegundos (000000..999999)
%p	AM o PM
%w	Día de la semana (0=Sunday..6=Saturday)
%U	Semana (00..53), donde el domingo es el primer día de la semana
%u	Semana (00..53), donde el lunes es el primer día de la semana
%V	Semana (01..53), donde el domingo es el primer día de la semana; usado con %X
%v	Semana (01..53), donde el lunes es el primer día de la semana; usado con %X
%%	Un '%' literal.

El resto de los caracteres se copian tal cual al resultado sin interpretación. El especificador de formato %f está disponible desde MySQL 4.1.1. Desde MySQL 3.23, se requiere el carácter '%' antes de los caracteres de especificación de formato. En versiones de MySQL más antiguas, '%' era opcional. El motivo por el

que los rangos de meses y días empiecen en cero es porque MySQL permite almacenar fechas incompletas como '2004-00-00', desde MySQL 3.23.

```
SELECT DATE_FORMAT('1997-10-04 22:23:00', '%W %M %Y');

SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H:%i:%s');

SELECT DATE_FORMAT('1997-10-04 22:23:00', '%D %y %a %d %m %b %j');

SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H %k %I %r %T %S %w');

SELECT DATE_FORMAT('1999-01-01', '%X %V');

SELECT DATE_FORMAT (FecNacimiento, '%W, %d-%m-%Y') FROM Empleados;

SELECT DATE_FORMAT (FecNacimiento, '%a, %d-%m-%Y') FROM Empleados;
```

DAYOFMONTH() / DAY()

DAYOFMONTH(date)

DAY(date)

Devuelve el día del mes para la fecha dada, en el rango de 1 a 31:

DAY() es un sinónimo de **DAYOFMONTH()**.

```
SELECT DAYOFMONTH('1998-02-03'), DAYOFMONTH(NOW()), DAY('1998-02-03'), DAY(NOW());
```

DAYNAME()

DAYNAME(date)

Devuelve el nombre del día de la semana para una fecha:

```
SELECT DAYNAME('1998-02-05'), DAYNAME(NOW());
```

DAYOFWEEK()

DAYOFWEEK(date)

Devuelve el índice del día de la semana para una fecha dada (1 = Sunday, 2 = Monday, ... 7 = Saturday). Estos valores de índice corresponden al estándar de ODBC.

```
SELECT DAYOFWEEK('1998-02-03'), DAYOFWEEK(NOW());
```

DAYOFYEAR()

DAYOFYEAR(date)

Devuelve el día del año para la fecha dada, en el rango de 1 a 366:

```
SELECT DAYOFYEAR('1998-05-03'), DAYOFYEAR(NOW());
```

EXTRACT()

EXTRACT(type FROM date)

La función **EXTRACT()** extrae partes de la fecha; los tipos permitidos de intervalos son los mismos que en **DATE_ADD()** o **DATE_SUB**().

```
SELECT EXTRACT(YEAR FROM "1999-07-02"), EXTRACT(MONTH FROM "1999-07-02"), EXTRACT(DAY FROM "1999-07-02");
```

```
SELECT EXTRACT(YEAR_MONTH FROM "1999-07-02 01:02:03"), EXTRACT(DAY_MINUTE FROM "1999-07-02 01:02:03");
```

```
SELECT EXTRACT(MICROSECOND FROM "2003-01-02 10:30:00.00123");
```

```
SELECT EXTRACT(YEAR FROM FecNacimiento), EXTRACT(MONTH FROM Fecnacimiento), EXTRACT(DAY FROM FecNacimiento)
FROM Empleados;
```

FROM_DAYS()

FROM_DAYS(N)

Dado un número de día N, devuelve un valor de fecha DATE:

```
SELECT FROM_DAYS(729669), FROM_DAYS(1), FROM_DAYS(1000)
```

FROM_DAYS() no está diseñada para usarse con valores anteriores al comienzo del calendario Gregoriano (1582), porque no tiene en cuenta los días que se perdieron cuando el calendario se cambió.

FROM_UNIXTIME()

FROM_UNIXTIME(unix_timestamp)

FROM_UNIXTIME(unix_timestamp, formato)

Devuelve una representación del argumento unix_timestamp como un valor en el formato 'YYYY-MM-DD HH:MM:SS' o YYYYMMDDHHMMSS, dependiendo de si la función se usa en un contexto de cadena o numérico:

```
SELECT FROM_UNIXTIME(875996580), FROM_UNIXTIME(875996580) + 0;
```

Si se proporciona un formato, el resultado se formatea de acuerdo con la cadena de formato. format puede contener los mismos especificadores que se listan en la entrada de la función **DATE_FORMAT():**

```
SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(), '%Y %D %M %h:%i:%s %x');
```

GET_FORMAT()

GET_FORMAT(DATE | TIME | TIMESTAMP, 'EUR' | 'USA' | 'JIS' | 'ISO' | 'INTERNAL')

Devuelve una cadena de formato. Esta función es frecuente en combinación con las funciones **DATE_FORMAT()** y **STR_TO_DATE().** Los tres posibles valores para el primer argumento y los cinco para el segundo implican quince posibles cadenas de

formato (para los especificadores usados, ver la tabla en la descripción de la función DATE_FORMAT()):

Llamada a función	Resultado
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(TIMESTAMP,'USA')	'%Y-%m-%d-%H.%i.%s'
GET_FORMAT(TIMESTAMP,'JIS')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(TIMESTAMP,'ISO')	'%Y-%m-%d %H:%i:%s'
GET_FORMAT(TIMESTAMP,'EUR')	'%Y-%m-%d-%H.%i.%s'
GET_FORMAT(TIMESTAMP,'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'

El formato ISO se refiere a ISO 9075, no a ISO 8601.

```
SELECT DATE_FORMAT('2003-10-03', GET_FORMAT(DATE, 'EUR')) ;
SELECT STR_TO_DATE('10.31.2003', GET_FORMAT(DATE, 'USA'));
SELECT DATE_FORMAT(FecNacimiento, GET_FORMAT(DATE, 'EUR'))FROM Empleados;
```

HOUR()

HOUR(time)

Devuelve la hora para time. El rango del valor retornado puede ser de 0 a 23 para valores de horas correspondientes al día:

Sin embargo, el rango de valores de TIME es mucho mayor, de modo que HOUR puede devolver valores mayores de 23:

```
SELECT HOUR('10:05:03'), HOUR('272:59:59');
```

LAST_DAY()

LAST_DAY(date)

Toma un valor fecha o fecha y hora y devuelve el valor correspondiente para el último día del mes. Devuelve NULL si el argumento no es válido.

```
SELECT LAST_DAY('2003-02-05'), LAST_DAY('2004-02-05'), LAST_DAY('2004-01-01 01:01:01'), LAST_DAY('2003-03-32');
```

MAKEDATE()

MAKEDATE(year, dayofyear)

Devuelve una fecha, dados los valores del año y de día del año. dayofyear debe ser mayor que 0 o el resultado será NULL.

```
SELECT MAKEDATE(2001,31), MAKEDATE(2001,32), MAKEDATE(2001,365), MAKEDATE(2004,365), MAKEDATE(2001,0), MAKEDATE(4,90);
```

MAKETIME()

MAKETIME(hour, minute, second)

Devuelve un valor de tiempo calculado a partir de los argumentos hour, minute y second.

```
SELECT MAKETIME(12,15,30);
```

MICROSECOND()

MICROSECOND(expr)

Devuelve los microsegundos a partir de una expresión tiempo o fecha y tiempo como un número en el rango de 0 a 999999.

```
SELECT MICROSECOND('12:00:00.123456'), MICROSECOND('1997-12-31 23:59:59.000010');
```

MINUTE()

MINUTE(time)

Devuelve el minuto para el tiempo time, en el rango de 0 a 59:

```
SELECT MINUTE('98-02-03 10:05:03'), MINUTE('98-02-03 10:85:03') ;
```

MONTH()

MONTH(date)

Devuelve el mes de una fecha, en el rango de 1 a 12:

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

MONTHNAME()

MONTHNAME(date)

Devuelve el nombre del mes para la fecha date:

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

QUARTER()

QUARTER(date)

Devuelve el cuarto del año para la fecha date, en el rango de 1 a 4:

```
SELECT QUARTER('98-04-01'), QUARTER(NOW()), QUARTER('2011-12-23);
```

SECOND()

SECOND(time)

Devuelve el segundo para el tiempo time, en el rango de 0 a 59:

```
SELECT SECOND('10:05:03');
```

SEC_TO_TIME()

SEC_TO_TIME(seconds)

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

```
SELECT SEC_TO_TIME(2378), SEC_TO_TIME(2378) + 0;
```

STR_TO_DATE()

STR_TO_DATE(str, format)

Esta es la función inversa de la función **DATE_FORMAT().** Toma una cadena str, y una cadena format, y devuelve un valor DATETIME. Los valores date, time o datetime contenidos en str deben ser dados en el formato indicado por format. Para ver los especificadores que pueden ser usados en format, ver la tabla en la descripción de la función **DATE_FORMAT().** El resto de los caracteres se toman tal cual, y no son interpretadas. Si str contiene una fecha, un valor tiempo o de fecha y tiempo ilegal, **STR_TO_DATE()** devuelve NULL.

```
SELECT STR_TO_DATE('03.10.2003 09.20', '%d.%m.%Y %H.%i'), STR_TO_DATE('10rap', '%crap'),

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

TIME()

TIME(expr)

Extrae la parte de la hora de la expresión expr del tipo tiempo o fecha y hora.

```
SELECT TIME('2003-12-31 01:02:03'), TIME('2003-12-31 01:02:03.000123');
```

TIMESTAMP()

TIMESTAMP(expr)

TIMESTAMP(expr,expr2)

Con un argumento, devuelve la expresión expr de fecha o fecha y hora como un valor fecha y tiempo. Con dos argumentos, suma la expresión de tiempo expr2 a la expresión de fecha o fecha y hora expr y devuelve un valor de fecha y tiempo.

```
SELECT TIMESTAMP('2003-12-31');

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

TIME_FORMAT()

TIME_FORMAT(time,format)

Se usa como la función **DATE_FORMAT**(), pero la cadena de formato sólo puede contener aquellos especificadores de formato que manejan horas, minutos y segundos. Otros especificadores producen un valor NULL o 0. Si el valor de tiempo contiene una parte de hora mayor que 23, los especificadores de formato de hora %H y %k producen un valor mayor que el usual de 0..23. Los otros especificadores de formato de hora producen el valor de hora módulo 12:

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

TIME_TO_SEC()

TIME_TO_SEC(time)

Devuelve el argumento time convertido en segundos:

```
SELECT TIME_TO_SEC('22:23:00'), TIME_TO_SEC('00:39:38');
```

TO_DAYS()

TO_DAYS(date)

Dada la fecha date, devuelve el número de días. Las fechas deben ser posteriores al año 1582; en caso contrario, el resultado no es seguro. Es la inversa de FROM_DAYS()

```
SELECT TO_DAYS(950501), TO_DAYS('2011-10-07');
```

TO_DAYS() no está diseñada para trabajar con valores anteriores a la implantación del calendario Gregoriano (1582), porque no tiene en cuenta los días perdidos cuando se instauró dicho calendario.

UNIX_TIMESTAMP()

UNIX_TIMESTAMP()

UNIX_TIMESTAMP(date)

Si es llamada sin argumentos, devuelve un timestamp Unix (segundos desde '1970-01-01 00:00:00' GMT) como un entero sin signo. Si es llamada con un argumento fecha, devuelve el valor del argumento como segundos desde '1970-01-01 00:00:00'

GMT. date debe ser una cadena DATE, una cadena DATETIME, un TIMESTAMP o un número en el formato YYMMDD o YYYYMMDD en hora local:

```
SELECT UNIX_TIMESTAMP(), UNIX_TIMESTAMP('1997-10-04 22:23:00');
```

Cuando se usa **UNIX_TIMESTAMP** en una columna *TIMESTAMP*, la función devuelve el valor interno timestamp directamente, sin la conversión implícita "string-to-Unix-timestamp". Si se usa una fecha fuera de rango a **UNIX_TIMESTAMP()** devuelve 0, pero hay que tener en cuenta que sólo se hace una comprobación básica (año en el margen 1970-2037, mes en 01-12 y día en 01-31). Si se quieren restar columnas **UNIX_TIMESTAMP(), puede desearse convertir el resultado a enteros con signo.**

UTC_DATE / UTC_DATE()

UTC_DATE

UTC_DATE()

Devuelve la fecha UTC actual como un valor en el formato 'YYYY-MM-DD' o YYYYMMDD, dependiendo de si se usa en un contexto de cadena o numérico:

```
SELECT UTC_DATE(), UTC_DATE() + 0
```

UTC_TIME / UTC_TIME()

UTC_TIME

UTC_TIME()

Devuelve la hora UTC actual como un valor en el formato 'HH:MM:SS' o HHMMSS, dependiendo de si se usa en un contexto de cadena o numérico:

```
SELECT UTC_TIME(), UTC_TIME() + 0;
```

UTC_TIMESTAMP / UTC_TIMESTAMP()

UTC_TIMESTAMP()

Devuelve la fecha y hora UTC actual como un valor en el formato 'YYYY-MM-DD HH:MM:SS' o YYYYMMDDHHMMSS, dependiendo de si se usa en un contexto de cadena o numérico:

```
SELECT UTC_TIMESTAMP(), UTC_TIMESTAMP() + 0;
```

WEEK()

WEEK(date [,mode])

Esta función devuelve el número de la semana para una fecha. El formato con dos argumentos permite especificar si la semana empieza en domingo o en lunes y si el valor de retorno debe estar en el rango 0-53 o 1-52. Cuando se omite el argumento de modo el valor por defecto usado es el de la variable del servidor default_week_format (o 0 en MySQL 4.0 o anterior). La tabla siguiente demuestra cómo trabaja el argumento mode:

Valor	Significado
0	La semana empieza en domingo; devuelve un valor en el rango 0 a 53; la semana 1 es la primera semana que empieza en este año
1	La semana empieza en lunes; devuelve un valor en el rango 0 a 53; la semana 1 es la primera semana que tiene más de 3 días en este año
2	La semana empieza en domingo; devuelve un valor en el rango 1 a 53; la semana 1 es la primera semana que empieza en este año
3	La semana empieza en lunes; devuelve un valor en el rango 1 a 53; la semana 1 es la primera semana que tenga más de tres días en este año
4	La semana empieza en domingo; devuelve un valor en el rango 0 a 53; la semana 1 es la primera semana que tenga más de tres días en este año
5	La semana empieza en lunes; devuelve un valor en el rango 0 a 53; la semana 1 es la primera semana que empiece en este año
6	La semana empieza en domingo; devuelve un valor en el rango 1 a 53; la semana 1 es la primera semana que tenga más de tres días en este año
7	La semana empieza en lunes; devuelve un valor en el rango 1 a 53; la semana 1 es la primera semana que empiece en este este año

El valor de modo 3 puede usarse desde MySQL 4.0.5. El valor de modo 4 y superiores pueden usarse desde MySQL 4.0.17.

```
SELECT WEEK('1998-02-20'), WEEK('1998-02-20',0), WEEK('1998-02-20',1), WEEK('1998-12-31',1),YEAR('2000-01-01'), WEEK('2000-01-01',0);
```

Se puede argumentar que MySQL debe devolver 52 en la función **WEEK()**, porque la fecha dada está en la semana 53 de 1999. Pero se ha decidido devolver 0 en su lugar ya que se prefiere que la función devuelva "el número de la semana en el año dado". Esto hace el uso de la función **WEEK()** function más fiable cuando se combina con otras funciones que extraen una parte de la fecha. Si se prefiere que el resultado sea evaluado con respecto al año que contiene el primer día de la semana para la fecha dada, se debe usar 2, 3, 6 ó 7 en el argumento mode.

```
SELECT WEEK('2000-01-01',2);
```

Alternativamente, se puede usar la función **YEARWEEK**():

```
SELECT WEEK('2000-01-01',2), YEARWEEK('2000-01-01', 2), MID(YEARWEEK('2000-01-01'),5,2);
```

WEEKDAY()

WEEKDAY(date)

Devuelve el índice del día de la semana para la fecha date (0 = Lunes, 1 = Martesy, ... 6 = Domingo):

```
SELECT WEEKDAY('1998-02-03 22:23:00'), WEEKDAY('2012-02-02');
```

WEEKOFYEAR()

WEEKOFYEAR(date)

Devuleve el número de semana según el calendario de la fecha dada como un número en el rango de 1 a 53.

```
SELECT WEEKOFYEAR('1998-02-20'), WEEKOFYEAR('2012-02-02');
```

YEAR()

YEAR(date)

Devuelve el año para una fecha, en el rango de 1000 a 9999:

```
SELECT YEAR('98-02-03'), YEAR(NOW());
```

YEARWEEK()

YEARWEEK(date)

YEARWEEK(date, start)

Devuelve el año y semana de una fecha. El argumento start trabaja exactamente igual que el argumento argument en la función **WEEK**(). El año en el resultado puede ser diferente del año en el argumento date para la primera y última semana del año:

```
SELECT YEARWEEK('1987-01-01'), YEARWEEK(NOW());
```

El número de semana es diferente que para la función **WEEK**(), que puede devolver el valor 0 para los argumentos opcionales 0 o 1, ya que **WEEK**().en ese caso, devuelve la semana en el contexto del año dado.

Funciones para cálculos con datos de fecha y hora

ADDDATE()

ADDDATE(date, INTERVAL expr type)

ADDDATE(expr, days)

Le suma a la fecha el valor **expr** que puede tener distintas inidades según sea el tipo. Por ejemplo, **expr** puede ser 2 y tipo DAY, eso significasumar dos días a la fecha. Cuando se invoca con el formato **INTERVAL** para el segundo argumento, **ADDDATE**() es sinonimo de **DATE_ADD**(). La función relacionada **SUBDATE**() es sinónimo de **DATE_SUB**().

```
SELECT DATE_ADD('1998-01-02', INTERVAL 31 DAY), ADDDATE('1998-01-02', INTERVAL 3 MONTH), DATE_ADD('1998-01-02', INTERVAL 5 YEAR);
```

Desde MySQL 4.1.1, se permite la segunda sintaxis, donde expr es una fecha o una expresión 'datetime' y days es el número de días a añadir a expr.

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

ADDTIME()

ADDTIME(expr1,expr2)

ADDTIME() suma dos expresiones, expr1 es del tipo fecha y hora o hora, (datetime o time) y expr2 del tipo hora (time).

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

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

DATEDIFF()

DATEDIFF(expr1,expr2)

DATEDIFF() devuelve el número de días entre la fecha de inicio **expr1** y la de final **expr2**. expr1 y expr2 son expresiones de tipo date o datetime. Sólo las partes correspondientes a la fecha de cada expresión se usan en los cáculos.

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

vDATE_ADD() / DATE_SUB()

DATE_ADD(date, INTERVAL expr type)

DATE_SUB(date, INTERVAL expr type)

Estas funciones realizan aritmética con fechas. Desde MySQL 3.23, INTERVAL expr type se permite en cualquiera de los lados del operador + si la expresión en el otro lado es un valor de tipo date o datetime. Para el operador -, INTERVAL expr type se permite sólo en el lado derecho, porque no tiene sentido restar un valor de tipo date o datetime desde un intervalo. (Ver ejemplos.) date es un valor de tipo DATETIME o DATE que especifica la fecha de comienzo. expr es una expresión que especifica un valor de intervalo a añadir o restar desde la fecha de comienzo. expr es una cadena; puede empezar con un '-' para intervalos negativos. type es una palabra clave que indica cómo debe interpretarse la expresión. La tabla siguiente muestra cómo se relacionan los argumentos type y expr:

Valor *type*	**Formato de *expr* esperado**
MICROSECOND	Microsegundos
SECOND	Segundos
MINUTE	Minutos
HOUR	Horas
DAY	Días
WEEK	Semanas
MONTH	Meses
QUARTER	Trimestres

YEAR	Años
SECOND_MICROSECOND	'Segundos:Microsegundos'
MINUTE_MICROSECOND	'Minutos:Microsegundos'
MINUTE_SECOND	'Minutos:Segundos'
HOUR_MICROSECOND	'Horas:Microsegundos'
HOUR_SECOND	'Horas:Minutos:Segundos'
HOUR_MINUTE	'Horas:Minutos
DAY_MICROSECOND	'Días.Microsegundos'
DAY_SECOND	'Días Horas:Minutos:Segundos'
DAY_MINUTE	'Días Horas:Minutos'
DAY_HOUR	'Días Horas'
YEAR_MONTH	'Años-Meses'

Los valores de tipo DAY_MICROSECOND, HOUR_MICROSECOND, MINUTE_MICROSECOND, SECOND_MICROSECOND, and MICROSECOND están permitidos desde MySQL 4.1.1. Los valores QUARTER y WEEK están permitidos desde MySQL 5.0.0. MySQL permite cualquier delimitador de puntuación en el formato de expr. Los mostrados en la tabla son los delimitadores propuestos. Si el argumento date es un valor DATE y los cálculos involucran sólo las partes de YEAR, MONTH y DAY (es decir, no las partes de hora), el resultado es un valor DATE. En otro caso, el resultado en un valor DATETIME:

```
SELECT '1997-12-31 23:59:59' + INTERVAL 1 SECOND;

SELECT INTERVAL 1 DAY + '1997-12-31';

SELECT '1998-01-01' - INTERVAL 1 SECOND;

SELECT DATE_ADD('1997-12-31 23:59:59', INTERVAL 1 SECOND);

SELECT DATE_ADD('1997-12-31 23:59:59', INTERVAL 1 DAY);

SELECT DATE_ADD('1997-12-31 23:59:59', INTERVAL '1:1' MINUTE_SECOND);

SELECT DATE_SUB('1998-01-01 00:00:00', INTERVAL '1 1:1:1' DAY_SECOND);

SELECT DATE_ADD('1998-01-01 00:00:00', INTERVAL '-1 10' DAY_HOUR);

SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);

SELECT DATE_ADD('1992-12-31 23:59:59.000002', INTERVAL '1.999999' SECOND_MICROSECOND);
```

Si se especifica un valor de intervalo demasiado corto (que no incluye todas las partes del intervalo que se esperan dada la palabra clave en tipo), MySQL asume que se han perdido las partes de la izquierda del valor de intervalo. Por ejemplo, si se especifica un tipo DAY_SECOND, se espera que el valor de expr contenga las

partes correspondientes a días, horas, minutos y segundos. Si se especifica un valor como '1:10', MySQL asume que las partes de días y horas se han perdido y el valor representa minutos y segundos. En otras palabras, '1:10' DAY_SECOND se interpreta del mismo modo que si se hubiese usado '1:10' MINUTE_SECOND. Esto es análogo al modo en que MySQL interpreta valores TIME que representan intervalos de tiempo en lugar de tiempo con respecto al día. Si se suma o resta desde un valor de fecha algo que contiene la parte de la hora, el resultado se convierte automáticamente a un valor de fecha y hora:

```
SELECT DATE_ADD('1999-01-01', INTERVAL 1 DAY);
```

```
SELECT DATE_ADD('1999-01-01', INTERVAL 1 HOUR);
```

Si se usan fechas mal formadas, el resultado es NULL. Si se suma MONTH, YEAR_MONTH o YEAR y la fecha resultado contiene un día que es mayor que el máximo para el nuevo mes, el día se ajusta al máximo para el mes nuevo:

```
SELECT DATE_ADD('1998-01-30', interval 1 month);
```

PERIOD_ADD()

PERIOD_ADD(P, N)

Añade N meses al periodo P (en el formato YYMM o YYYYMM). Devuelve un valor en el formato YYYYMM. El argumento periodo P no es un valor de fecha:

```
SELECT PERIOD_ADD(9801,2);
```

PERIOD_DIFF()

PERIOD_DIFF(P1,P2)

Devuelve el número de meses entre los periodos P1 y P2. P1 y P2 deben estar en el formato YYMM o YYYYMM. Los argumentos periodo P1 y P2 no son valores de fecha:

```
SELECT PERIOD_DIFF(9802,199703);
```

SUBDATE()

SUBDATE(date, INTERVAL expr type)

SUBDATE(expr,days)

Cuando se invoca con el formato INTERVAL en el segundo argumento, **SUBDATE**() es un sinónimo de **DATE_SUB().**

```
SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY);
```

```
SELECT SUBDATE('1998-01-02', INTERVAL 31 DAY);
```

Desde la versión MySQL 4.1.1, se permite la segunda sintaxis, donde expr es una expresión de fecha o de fecha y hora y days es el número de días a restar desde expr.

```
SELECT SUBDATE('1998-01-02 12:00:00', 31);
```

SUBTIME()

SUBTIME(expr, expr2)

SUBTIME() resta expr2 de expr y devuelve el resultado. expr es una expresión fecha o fecha y hora, y expr2 es una expresión tiempo.

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

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

TIMEDIFF()

TIMEDIFF(expr, expr2)

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

```
SELECT TIMEDIFF('2000:01:01 00:00:00', '2000:01:01 00:00:00.000001');

SELECT TIMEDIFF('1997-12-31 23:59:59.000001','1997-12-30 01:01:01.000002');
```

TIMESTAMPADD()

TIMESTAMPADD(interval,int_expr,datetime_expr)

Suma la expresión entera int_expr a la expresión de fecha o fecha y hora datetime_expr. Las unidades para int_expr se toman del argumento interval, que debe ser uno de los siguiente: FRAC_SECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER o YEAR. El valor de interval value puede ser especificado usando una de las palabras clave mostradas, o con un prefijo de SQL_TSI_. Por ejemplo, DAY o SQL_TSI_DAY son ambas legales.

```
SELECT TIMESTAMPADD(MINUTE,1,'2003-01-02');

SELECT TIMESTAMPADD(WEEK,1,'2003-01-02');
```

TIMESTAMPDIFF()

TIMESTAMPDIFF(interval,datetime_expr1,datetime_expr2)

Devuelve la diferencia entera entre las expresiones fecha o fecha y hora datetime_expr1 y datetime_expr2. Las unidades para el resultado vienen dadas por el argumento interval. Los valores legales para interval son los mismos que se mencionan en la descripción de la función TIMESTAMPADD().

```
SELECT TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01');

SELECT TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01');
```