Данная статья описывает синтаксис функций для работы с датами и временем в MySQL

Ниже представлен пример, в котором используются функции даты. Приведенный запрос выбирает все записи с величиной date_col в течение последних 30 дней:

mysql> SELECT something FROM tbl_name WHERE TO_DAYS(NOW()) - TO_DAYS(date_col) SELECT DAYOFWEEK('1998-02-03'); -> 3

-

WEEKDAY(date)

Возвращает индекс дня недели для аргумента date (0 =понедельник, 1 = вторник, ... 6 = воскресенье):

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

-

DAYOFMONTH(date)

Возвращает порядковый номер дня месяца для аргумента date в диапазоне от 1 до 31:

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

-

DAYOFYEAR(date)

Возвращает порядковый номер дня года для аргумента date в диапазоне от 1 до 366:

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

-

MONTH(date)

Возвращает порядковый номер месяца в году для аргумента date в диапазоне от 1 до 12:

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

-

DAYNAME(date)

Возвращает название дня недели для аргумента date:

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

-

MONTHNAME(date)

Возвращает название месяца для аргумента date:

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

-

QUARTER(date)

Возвращает номер квартала года для аргумента date в диапазоне от 1 до 4:

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

-

WEEK(date) , WEEK(date,first)

При наличии одного аргумента возвращает порядковый номер недели в году для date в диапазоне от 0 до 53 (да, возможно начало 53-й недели) для регионов, где воскресенье считается первым днем недели. Форма WEEK() с двумя аргументами позволяет уточнить, с какого дня начинается неделя - с воскресенья или с понедельника. Результат будет в пределах 0-53 или 1-52.

Вот как работает второй аргумент:

Величина	Означает
0	Неделя начинается с воскресенья; возвращаемое значение - в промеж
1	Неделя начинается с понедельника; возвращаемое значение - в проме
2	Неделя начинается с воскресенья; возвращаемое значение - в промеж
3	Неделя начинается с понедельника; возвращаемое значение - в проме

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

Примечание: в версии 4.0 функция WEEK(#,0) была изменена с целью соответствия календарю США.

Заметьте, если неделя является последней неделей прошлого года, MySQL вернет 0 если вы не указали 2 или 3 как опциональный аргумент:

mysql> SELECT YEAR('2000-01-01'), WEEK('2000-01-01',0); -> 2000, 0 mysql> SELECT WEEK('2000-01-01',2); -> 52

Можно считать, что MySQL должен вернуть 52, так как данная дата и является 52-ой неделей года 1999. Мы решили возвращать 0, так как мы хотим, чтобы функция давала "номер недели в указанном году". Это делает функциюWEEK() более надежной при использовании совместно с другими функциями, которые вычисляют части дат.

Если вам все же важно уточнить корректную неделю в году, тогда вы можете использовать 2 или 3 как опциональный аргумент или использовать YEARWEEK()

```
mysql> SELECT YEARWEEK('2000-01-01');          -> 199952  mysql> SELECT
MID(YEARWEEK('2000-01-01'),5,2);          -> 52
```

-

YEAR(date)

Возвращает год для аргумента date в диапазоне от 1000 до 9999:

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

-

YEARWEEK(date) , YEARWEEK(date,first)

Возвращает год и неделю для аргумента date. Второй аргумент в данной функции работает подобно второму аргументу в функции WEEK(). Следует учитывать, что год может отличаться от указанного в аргументе date для первой и последней недель года:

```
mysql> SELECT YEARWEEK('1987-01-01');          -> 198653
```

Обратите внимание, что номер недели отличается от того, который возвращает функция WEEK() (0), будучи вызванной с опциональным аргументом 0 или 1. Это потому, что WEEK() возвращает номер недели именно в указанном году.

-

HOUR(time)

Возвращает час для аргумента time в диапазоне от 0 до 23:

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

-

MINUTE(time)

Возвращает количество минут для аргумента time в диапазоне от 0 до 59:

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

-

SECOND(time)

Возвращает количество секунд для аргумента time в диапазоне от 0 до 59:

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

-

PERIOD_ADD(P,N)

Добавляет N месяцев к периоду P (в формате YYMM или YYYYMM). Возвращает величину в формате YYYYMM. Следует учитывать, что аргумент периода P не является значением даты:

```
mysql> SELECT PERIOD_ADD(9801,2);          -> 199803
```

-

PERIOD_DIFF(P1,P2)

Возвращает количество месяцев между периодами P1 и P2. P1 и P2 должны быть в формате YYMM или YYYYMM. Следует учитывать, что аргументы периода P1 и P2 не являются значениями даты:

```
mysql> SELECT PERIOD_DIFF(9802,199703);          -> 11
```

-

DATE_ADD(date,INTERVAL expr type) , DATE_SUB(date,INTERVAL expr type)
, ADDDATE(date,INTERVAL expr type) , SUBDATE(date,INTERVAL expr type)

Данные функции производят арифметические действия над датами. Обе являются нововведением версии MySQL 3.22. Функции ADDDATE() и SUBDATE() - синонимы для DATE_ADD() и DATE_SUB(). В версии MySQL 3.23 вместо функций DATE_ADD() и DATE_SUB() можно использовать операторы + и -, если выражение с правой стороны представляет собой столбец типа DATE или DATETIME (см. пример ниже). Аргумент date

является величиной типаDATETIME или DATE, задающей начальную дату.

Выражение expr задает величину интервала, который следует добавить к начальной дате или вычесть из начальной даты. Выражение expr представляет собой строку, которая может начинаться с - для отрицательных значений интервалов. Ключевое слово type показывает, каким образом необходимо интерпретировать данное выражение. Вспомогательная функция EXTRACT(type FROM date) возвращает интервал указанного типа (type) из значения даты. В следующей таблице показана взаимосвязь аргументов type и expr:

Значение	Type	**Ожидаемый формат**	expr

SECOND	SECONDS
MINUTE	MINUTES
HOUR	HOURS
DAY	DAYS
MONTH	MONTHS
YEAR	YEARS
MINUTE_SECOND	"MINUTES:SECONDS"
HOUR_MINUTE	"HOURS:MINUTES"
DAY_HOUR	"DAYS HOURS"
YEAR_MONTH	"YEARS-MONTHS"
HOUR_SECOND	"HOURS:MINUTES:SECONDS"
DAY_MINUTE	"DAYS HOURS:MINUTES"
DAY_SECOND	"DAYS HOURS:MINUTES:SECONDS"

В MySQL формат выражения expr допускает любые разделительные знаки. Разделители, представленные в данной таблице, приведены в качестве примеров. Если аргумент date является величиной типа DATE и предполагаемые вычисления включают в себя только части YEAR, MONTH, и DAY (т.е. не содержат временной части TIME), то результат представляется величиной типа DATE. В других случаях результат представляет собой величину DATETIME:

```
mysql> SELECT "1997-12-31 23:59:59" + INTERVAL 1 SECOND;
-> 1998-01-01 00:00:00
mysql> SELECT INTERVAL 1 DAY + "1997-12-31";
-> 1998-01-01
mysql> SELECT "1998-01-01" - INTERVAL 1 SECOND;
-> 1997-12-31 23:59:59
mysql> SELECT DATE_ADD("1997-12-31 23:59:59", INTERVAL 1 SECOND);
-> 1998-01-01 00:00:00
mysql> SELECT DATE_ADD("1997-12-31 23:59:59", INTERVAL 1 DAY);
-> 1998-01-01 23:59:59
mysql> SELECT DATE_ADD("1997-12-31 23:59:59", INTERVAL "1:1"
```

MINUTE_SECOND); -> 1998-01-01 00:01:00 mysql> SELECT DATE_SUB("1998-01-01 00:00:00", INTERVAL "1 1:1:1" DAY_SECOND); -> 1997-12-30 22:58:59 mysql> SELECT DATE_ADD("1998-01-01 00:00:00", INTERVAL "-1 10" DAY_HOUR); -> 1997-12-30 14:00:00 mysql> SELECT DATE_SUB("1998-01-02", INTERVAL 31 DAY); -> 1997-12-02

Если указанный интервал слишком короткий (т.е. не включает все части интервала, ожидаемые при заданном ключевом слове type), то MySQL предполагает, что опущены крайние слева части интервала. Например, если указан аргумент type в виде DAY_SECOND, то ожидаемое выражение expr должно иметь следующие части: дни, часы, минуты и секунды. Если в этом случае указать значение интервала в виде "1:10", то MySQL предполагает, что опущены дни и часы, а данная величина включает только минуты и секунды. Другими словами, сочетание "1:10" DAY_SECOND интерпретируется как эквивалент "1:10" MINUTE_SECOND. Аналогичным образом в MySQL интерпретируются и значения TIME - скорее как представляющие прошедшее время, чем как время дня. Следует учитывать, что при операциях сложения или вычитания с участием величины DATE и выражения, содержащего временную часть, данная величина DATE будет автоматически конвертироваться в величину типа DATETIME:

mysql> SELECT DATE_ADD("1999-01-01", INTERVAL 1 DAY); -> 1999-01-02 mysql> SELECT DATE_ADD("1999-01-01", INTERVAL 1 HOUR); -> 1999-01-01 01:00:00

При использовании некорректных значений дат результат будет равен NULL. Если при суммировании MONTH, YEAR_MONTH или YEAR номер дня в результирующей дате превышает максимальное количество дней в новом месяце, то номер дня результирующей даты принимается равным последнему дню нового месяца:

mysql> SELECT DATE_ADD('1998-01-30', INTERVAL 1 MONTH); -> 1998-02-28

Из предыдущего примера видно, что слово INTERVAL и ключевое слово type не являются регистро-зависимыми.

-

EXTRACT(type FROM date)

Типы интервалов для функции EXTRACT() используются те же, что и для функций DATE_ADD() или DATE_SUB(), но EXTRACT() производит скорее извлечение части из значения даты, чем выполнение арифметических действий.

mysql> SELECT EXTRACT(YEAR FROM "1999-07-02"); -> 1999 mysql> SELECT EXTRACT(YEAR_MONTH FROM "1999-07-02 01:02:03"); -> 199907 mysql> SELECT EXTRACT(DAY_MINUTE FROM "1999-07-02 01:02:03"); -> 20102

-

TO_DAYS(date)

функция возвращает номер дня для даты, указанной в аргументе date, (количество дней, прошедших с года 0):

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

Функция TO_DAYS() не предназначена для использования с величинами, предшествующими введению григорианского календаря (1582), поскольку не учитывает дни, утерянные при изменении календаря.

-

FROM_DAYS(N)

Возвращает величину DATE для заданного номера дня N:

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

Функция FROM_DAYS() не предназначена для использования с величинами, предшествующими введению григорианского календаря (1582), поскольку она не учитывает дни, утерянные при изменении календаря.

-

DATE_FORMAT(date,format)

Форматирует величину date в соответствии со строкой format. В строке format могут использоваться следующие определители:

Определитель	Описание
%M	Название месяца (январь...декабрь)
%W	Название дня недели (воскресенье...суббота)
%D	День месяца с английским суффиксом (0st, 1st, 2nd, 3rd и т.д.)
%Y	Год, число, 4 разряда
%y	Год, число, 2 разряда
%X	Год для недели, где воскресенье считается первым днем недели, числ
%x	Год для недели, где воскресенье считается первым днем недели, числ
%a	Сокращенное наименование дня недели (Вс...Сб)
%d	День месяца, число (00..31)
%e	День месяца, число (0..31)
%m	Месяц, число (00..12)
%c	Месяц, число (0..12)
%b	Сокращенное наименование месяца (Янв...Дек)
%j	День года (001..366)
%H	Час (00..23)
%k	Час (0..23)
%h	Час (01..12)
%I	Час (01..12)
%l	Час (1..12)
%i	Минуты, число (00..59)
%r	Время, 12-часовой формат (hh:mm:ss [AP]M)
%T	Время, 24-часовой формат (hh:mm:ss)
%S	Секунды (00..59)
%s	Секунды (00..59)
%p	AM или PM
%w	День недели (0=воскресенье..6=суббота)
%U	Неделя (00..53), где воскресенье считается первым днем недели
%u	Неделя (00..53), где понедельник считается первым днем недели
%V	Неделя (01..53), где воскресенье считается первым днем недели. Испо
%v	Неделя (01..53), где понедельник считается первым днем недели. Испо
%%	Литерал '%'.

Все другие символы просто копируются в результирующее выражение без интерпретации:

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

В MySQL 3.23 символ '%' должен предшествовать символам определителя формата. В более ранних версиях MySQL символ '%' необязателен.

Причина того, что промежутки для месяца и дня начинаются с нуля заключается в том, что MySQL позволяет использовать неполные даты, такие как '2004-00-00', начиная с MySQL 3.23.

-

TIME_FORMAT(time,format)

Данная функция используется аналогично описанной выше функции DATE_FORMAT(), но строка format может содержать только те определители формата, которые относятся к часам, минутам и секундам. При указании других определителей будет выдана величина NULL или 0.

-

CURDATE() , CURRENT_DATE

Возвращает сегодняшнюю дату как величину в формате YYYY-MM-DD или YYYYMMDD, в зависимости от того, в каком контексте используется функция - в строковом или числовом:

mysql> SELECT CURDATE(); -> '1997-12-15' mysql> SELECT CURDATE() + 0; -> 19971215

-

CURTIME() , CURRENT_TIME

Возвращает текущее время как величину в формате HH:MM:SS или HHMMS, в зависимости от того, в каком контексте используется функция - в строковом или числовом:

mysql> SELECT CURTIME(); -> '23:50:26' mysql> SELECT CURTIME() + 0; -> 235026

-

NOW() , SYSDATE() , CURRENT_TIMESTAMP

Возвращает текущую дату и время как величину в формате YYYY-MM-DD HH:MM:SS или YYYYMMDDHHMMSS, в зависимости от того, в каком контексте используется функция - в строковом или числовом:

mysql> SELECT NOW(); -> '1997-12-15 23:50:26' mysql> SELECT NOW() + 0; -> 19971215235026

Заметьте, что NOW() вычисляется только единожды для каждого запроса, а именно - в начале его выполнения. Это позволяет быть уверенным в том, что множественные ссылки на NOW() в рамках одного запроса дадут одно и то же значение.

-

UNIX_TIMESTAMP() , UNIX_TIMESTAMP(date)

При вызове данной функции без аргумента она возвращает временную метку UNIX_TIMESTAMP (секунды с 1970-01-01 00:00:00 GMT) как беззнаковое целое число. Если функция UNIX_TIMESTAMP() вызывается с аргументом date, она возвращает величину аргумента как количество секунд с 1970-01-01 00:00:00 GMT. Аргумент date может представлять собой строку типа DATE, строку DATETIME, величину типа TIMESTAMP или число в формате YYMMDD илиYYYYMMDD местного времени:

mysql> SELECT UNIX_TIMESTAMP(); -> 882226357 mysql> SELECT UNIX_TIMESTAMP('1997-10-04 22:23:00'); -> 875996580

При использовании функции UNIX_TIMESTAMP в столбце TIMESTAMP эта функция будет возвращать величину внутренней временной метки непосредственно, без подразумеваемого преобразования строки во временную метку (``string-to-unix-timestamp''). Если заданная дата выходит за пределы допустимого диапазона, то функция UNIX_TIMESTAMP() возвратит 0, но следует учитывать, что выполняется только базовая проверка (год 1970-2037, месяц 01-12, день 01-31). Если необходимо выполнить вычитание столбцов UNIX_TIMESTAMP(), результат можно преобразовать к целым числам со знаком.

-

FROM_UNIXTIME(unix_timestamp)

Возвращает представление аргумента unix_timestamp как величину в формате YYYY-MM-DD HH:MM:SS или YYYYMMDDHHMMSS, в зависимости от того, в каком контексте используется функция - в строковом или числовом:

mysql> SELECT FROM_UNIXTIME(875996580); -> '1997-10-04 22:23:00' mysql> SELECT FROM_UNIXTIME(875996580) + 0; -> 19971004222300

-

FROM_UNIXTIME(unix_timestamp,format)

Возвращает строковое представление аргумента unix_timestamp, отформатированное в соответствии со строкой format. Строка format может содержать те же определители, которые перечислены в описании для функцииDATE_FORMAT():

mysql> SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(), '%Y %D %M %h:%i:%s %x');
-> '1997 23rd December 03:43:30 1997'

-

SEC_TO_TIME(seconds)

Возвращает аргумент seconds, преобразованный в часы, минуты и секунды, как величину в формате HH:MM:SS или HHMMSS, в зависимости от того, в каком контексте используется функция - в строковом или числовом:

mysql> SELECT SEC_TO_TIME(2378); -> '00:39:38' mysql> SELECT SEC_TO_TIME(2378) + 0; -> 3938

-

TIME_TO_SEC(time)

Возвращает аргумент time, преобразованный в секунды:

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