# Лабораторная работа 17. Данные типа даты и времени MySQL

Ссылка: https://phpclub.ru/mysql/doc/date-and-time-types.html

Цель работы: научиться обработке данных типа даты и времени в СУБД MySQL.

Теоретическая часть.

Существуют следующие типы данных даты и времени: DATETIME, DATE, TIMESTAMP, TIME и YEAR. Каждый из них имеет интервал допустимых значений, а также значение «ноль», которое используется, когда пользователь вводит действительно недопустимое значение. Отметим, что MySQL позволяет хранить некоторые не вполне достоверные значения даты, например 1999-11-31. Для ускорения проверки правильности даты MySQL только проверяет, находится ли месяц в интервале 0-12 и день в интервале 0-31. Данные интервалы начинаются с 0, это сделано для того, чтобы обеспечить для MySQL возможность хранить в столбцах DATE или DATETIME даты, в которых день или месяц равен нулю. Эта возможность особенно полезна для приложений, которые предполагают хранение даты рождения - здесь не всегда известен день или месяц рождения. В таких случаях дата хранится просто в виде 1999-00-00 или 1999-01-00 (при этом не следует рассчитывать на то, что для подобных дат функции DATE\_SUB() или DATE\_ADD дадут правильные значения).

Ниже приведены некоторые общие соображения, полезные при работе с типами данных даты и времени:

MySQL извлекает значения для данного типа даты или времени только в стандартном формате, но в то же время пытается интерпретировать разнообразные форматы, которые могут поступать от пользователей (например, когда задается величина, которой следует присвоить тип даты или времени или сравнить со значением, имеющим один из этих типов). Тем не менее, поддерживаются только форматы, описанные в следующих разделах. Предполагается, что пользователь будет вводить допустимые значения величин, так как использование величин в других форматах может дать непредсказуемые результаты.

Хотя MySQL пытается интерпретировать значения в нескольких форматах, во всех случаях ожидается, что крайним слева будет раздел значения даты, содержащий год. Даты должны задаваться в порядке год-месяц-день (например, '98-09-04'), а не в порядке месяц-день-год или день-месяц-год, т.е. не так, как их обычно записывают (например '09-04-98', '04-09-98').

MySQL автоматически преобразует значение, имеющее тип даты или времени, в число, если данная величина используется в числовом контексте, и наоборот.

Значение, имеющее тип даты или времени, которое выходит за границы установленного интервала или является недопустимым для этого типа данных (см. начало раздела), преобразуется в значение «ноль» для данного типа. (Исключение составляют выходящие за границы установленного интервала величины типа TIME, которые усекаются до соответствующей граничной точки заданного интервала TIME). В следующей таблице представлены форматы значения «ноль» для каждого из типов столбцов:

|  |  |
| --- | --- |
| Тип столбца  | Значение «Ноль»  |
| DATETIME  | '0000-00-00 00:00:00'  |
| DATE  | '0000-00-00'  |
| TIMESTAMP  | 00000000000000 (длина зависит от количества выводимых символов)  |
| TIME  | '00:00:00'  |
| YEAR  | 0000  |

Значения «ноль» - особые. Для их хранения или ссылок на них можно явно применять представленные в таблице значения, а можно использовать `0' или `0', что легче в написании.

Значения «ноль» даты или времени при использовании MyODBC автоматически конвертируются в NULL, так как ODBC не оперирует с подобными величинами.

6.2.2.2 Типы данных DATETIME, DATE и TIMESTAMP

Типы DATETIME, DATE и TIMESTAMP являются родственными типами данных. В данном разделе описаны их свойства, общие черты и различия.

Тип данных DATETIME используется для величин, содержащих информацию как о дате, так и о времени. MySQL извлекает и выводит величины DATETIME в формате 'YYYY-MM-DD HH:MM:SS'. Поддерживается диапазон величин от '1000-01-01 00:00:00' до '9999-12-31 23:59:59'. (''поддерживается'' означает, что хотя величины с более ранними временными значениями, возможно, тоже будут работать, но нет гарантии того, что они будут правильно храниться и отображаться).

Тип DATE используется для величин с информацией только о дате, без части, содержащей время. MySQL извлекает и выводит величины DATE в формате 'YYYY-MM-DD'. Поддерживается диапазон величин от '1000-01-01' до '9999-12-31'.

Тип столбца TIMESTAMP обеспечивает тип представления данных, который можно использовать для автоматической записи текущих даты и времени при выполнении операций INSERT или UPDATE. При наличии нескольких столбцов типа TIMESTAMP только первый из них обновляется автоматически.

Автоматическое обновление первого столбца с типом TIMESTAMP происходит при выполнении любого из следующих условий:

Столбец не указан явно в команде INSERT или LOAD DATA INFILE.

Столбец не указан явно в команде UPDATE, и при этом изменяется величина в некотором другом столбце (следует отметить, что команда UPDATE, устанавливающая столбец в то же самое значение, которое было до выполнения команды, не вызовет обновления столбца TIMESTAMP, поскольку в целях повышения производительности MySQL игнорирует подобные обновления при установке столбца в его текущее значение).

Величина в столбце TIMESTAMP явно установлена в NULL.

Для остальных (кроме первого) столбцов типа TIMESTAMP также можно задать установку в значение текущих даты и времени. Для этого необходимо просто установить столбец в NULL или в NOW().

Любой столбец типа TIMESTAMP (даже первый столбец данного типа) можно установить в значение, отличное от текущих даты и времени. Это делается путем явной установки его в желаемое значение. Данное свойство можно использовать, например, если необходимо установить столбец TIMESTAMP в значение текущих даты и времени при создании строки, а при последующем обновлении этой строки значение столбца не должно изменяться.

Пусть MySQL автоматически установит значение столбца с типом TIMESTAMP при создании данной строки. Столбец будет установлен в исходное состояние со значением текущих даты и времени.

При выполнении последующих обновлений других столбцов в данной строке необходимо явно установить столбец TIMESTAMP в его текущее значение.

Однако, с другой стороны, для этих целей, возможно, будет проще использовать столбец DATETIME. При создании строки его следует инициализировать функцией NOW() и оставить в покое при последующих обновлениях.

Величины типа TIMESTAMP могут принимать значения от начала 1970 года до некоторого значения в 2037 году с разрешением в одну секунду. Эти величины выводятся в виде числовых значений.

Формат данных, в котором MySQL извлекает и показывает величины TIMESTAMP, зависит от количества показываемых символов. Это проиллюстрировано в приведенной ниже таблице. Полный формат TIMESTAMP составляет 14 десятичных разрядов, но можно создавать столбцы типа TIMESTAMP и с более короткой строкой вывода:

|  |  |
| --- | --- |
| Тип столбца  | Формат вывода  |
| TIMESTAMP(14)  | YYYYMMDDHHMMSS  |
| TIMESTAMP(12)  | YYMMDDHHMMSS  |
| TIMESTAMP(10)  | YYMMDDHHMM  |
| TIMESTAMP(8)  | YYYYMMDD  |
| TIMESTAMP(6)  | YYMMDD  |
| TIMESTAMP(4)  | YYMM  |
| TIMESTAMP(2)  | YY  |

Независимо от размера выводимого значения размер данных, хранящихся в столбцах типа TIMESTAMP, всегда один и тот же. Чаще всего используется формат вывода с 6, 8, 12 или 14 десятичными знаками. При создании таблицы можно указать произвольный размер выводимых значений, однако если этот размер задать равным 0 или превышающим 14, то будет использоваться значение 14. Нечетные значения размеров в интервале от 1 до 13 будут приведены к ближайшему большему четному числу.

Величины DATETIME, DATE и TIMESTAMP могут быть заданы любым стандартным набором форматов:

Как строка в формате 'YYYY-MM-DD HH:MM:SS' или в формате 'YY-MM-DD HH:MM:SS'. Допускается «облегченный» синтаксис - можно использовать любой знак пунктуации в качестве разделительного между частями разделов даты или времени. Например, величины '98-12-31 11:30:45', '98.12.31 11+30+45', '98/12/31 11\*30\*45' и '98@12@31 11^30^45' являются эквивалентными.

Как строка в формате 'YYYY-MM-DD' или в формате 'YY-MM-DD'. Здесь также допустим «облегченный» синтаксис. Например, величины '98-12-31', '98.12.31', '98/12/31' и '98@12@31' являются эквивалентными.

Как строка без разделительных знаков в формате 'YYYYMMDDHHMMSS' или в формате 'YYMMDDHHMMSS', при условии, что строка понимается как дата. Например, величины '19970523091528' и '970523091528' можно интерпретировать как '1997-05-23 09:15:28', но величина '971122129015' является недопустимой (значение раздела минут является абсурдным) и преобразуется в '0000-00-00 00:00:00'.

Как строка без разделительных знаков в формате 'YYYYMMDD' или в формате 'YYMMDD', при условии, что строка интерпретируется как дата. Например, величины '19970523' и '970523' можно интерпретировать как '1997-05-23', но величина '971332' является недопустимой (значения разделов месяца и дня не имеют смысла) и преобразуется в '0000-00-00'.

Как число в формате YYYYMMDDHHMMSS или в формате YYMMDDHHMMSS, при условии, что число интерпретируется как дата. Например, величины 19830905132800 и 830905132800 интерпретируются как '1983-09-05 13:28:00'.

Как число в формате YYYYMMDD или в формате YYMMDD, при условии, что число интерпретируется как дата. Например, величины 19830905 и 830905 интерпретируются как '1983-09-05'.

Как результат выполнения функции, возвращающей величину, приемлемую в контекстах типов данных DATETIME, DATE или TIMESTAMP (например, функции NOW() или CURRENT\_DATE).

Недопустимые значения величин DATETIME, DATE или TIMESTAMP преобразуются в значение «ноль» соответствующего типа величин ('0000-00-00 00:00:00', '0000-00-00', или 00000000000000).

Для величин, представленных как строки, содержащие разделительные знаки между частями даты, нет необходимости указывать два разряда для значений месяца или дня, меньших, чем 10. Так, величина '1979-6-9' эквивалентна величине '1979-06-09'. Аналогично, для величин, представленных как строки, содержащие разделительные знаки внутри обозначения времени, нет необходимости указывать два разряда для значений часов, минут или секунд, меньших, чем 10.

Величины, определенные как числа, должны иметь 6, 8, 12, или 14 десятичных разрядов. Предполагается, что число, имеющее 8 или 14 разрядов, представлено в форматах YYYYMMDD или YYYYMMDDHHMMSS соответственно, причем год указан в первых четырех разрядах. Если же длина числа 6 или 12 разрядов, то предполагаются соответственно форматы YYMMDD или YYMMDDHHMMSS, где год указан в первых двух разрядах. Числа, длина которых не соответствует ни одному из описанных вариантов, интерпретируются как дополненные спереди нулями до ближайшей вышеуказанной длины.

Величины, представленные строками без разделительных знаков, интерпретируются с учетом их длины согласно приведенным далее правилам. Если длина строки равна 8 или 14 символам, то предполагается, что год задан первыми четырьмя символами. В противном случае предполагается, что год задан двумя первыми символами. Строка интерпретируется слева направо, при этом определяются значения для года, месяца, дня, часов, минут и секунд для всех представленных в строке разделов. Это означает, что строка с длиной меньше, чем 6 символов, не может быть использована. Например, если задать строку вида '9903', полагая, что это будет означать март 1999 года, то MySQL внесет в таблицу «нулевую» дату. Год и месяц в данной записи равны 99 и 03 соответственно, но раздел, представляющий день, пропущен (значение равно нулю), поэтому в целом данная величина не является достоверным значением даты.

При хранении допустимых величин в столбцах типа TIMESTAMP используется полная точность, указанная при их задании, независимо от количества выводимых символов. Это свойство имеет несколько следствий:

Необходимо всегда указывать год, месяц и день даже для типов TIMESTAMP(4) или TIMESTAMP(2). В противном случае задаваемая величина не будет допустимым значением даты и будет храниться как 0.

При увеличении ширины узкого столбца TIMESTAMP путем использования команды ALTER TABLE будет выводиться ранее «скрытая» информация.

И аналогично, при сужении столбца TIMESTAMP хранимая информация не будет потеряна, если не принимать во внимание, что при выводе информации будет выдаваться меньше.

Хотя величины TIMESTAMP хранятся с полной точностью, непосредственно может работать с этим исходным хранимым значением величины только функция UNIX\_TIMESTAMP(). Остальные функции оперируют форматированными значениями извлеченной величины. Это означает, что нельзя использовать такие функции, как HOUR() или SECOND(), пока соответствующая часть величины TIMESTAMP не будет включена в ее форматированное значение. Например, раздел HH столбца TIMESTAMP не будет выводиться, пока количество выводимых символов не станет по меньшей мере равным 10, так что попытки использовать HOUR() для более коротких величин TIMESTAMP приведут к бессмысленным результатам.

Величины одного типа даты можно в ряде случаев присвоить объекту другого типа даты. Однако при этом возможны некоторое изменение величины или потеря информации.

Если присвоить значение типа DATE объекту DATETIME или TIMESTAMP, то в результирующей величине «временная»' часть будет установлена в '00:00:00', так как величина DATE не содержит информации о времени.

Если присвоить значение типа DATE, DATETIME или TIMESTAMP объекту DATE, то «временная» часть в результирующей величине будет удалена, так как тип DATE не включает информацию о времени.

Несмотря на то что все величины DATETIME, DATE и TIMESTAMP могут быть указаны с использованием одного и того же набора форматов, следует помнить, что указанные типы имеют разные интервалы допустимых значений. Например, величины типа TIMESTAMP не могут иметь значения даты более ранние, чем относящиеся к 1970 году или более поздние, чем относящиеся к 2037 году. Это означает, что такая дата, как '1968-01-01', будучи разрешенной для величины типа DATETIME или DATE, недопустима для величины типа TIMESTAMP и будет преобразована в 0 при присвоении этому объекту.

Задавая величины даты, следует иметь в виду некоторые особенности.

Упрощенный формат, который допускается для величин, заданных строками, может ввести в заблуждение. Например, такая величина, как '10:11:12', благодаря разделителю `:' могла бы оказаться величиной времени, но, используемая в контексте даты, она будет интерпретирована как год '2010-11-12'. В то же время величина '10:45:15' будет преобразована в '0000-00-00', так как для месяца значение '45' недопустимо.

Сервер MySQL выполняет только первичную проверку истинности даты: дни 00-31, месяцы 00-12, года 1000-9999. Любая дата вне этого диапазона преобразуется в 0000-00-00. Следует отметить, что, тем не менее, при этом не запрещается хранить неверные даты, такие как 2002-04-31. Это позволяет веб-приложениям сохранять данные форм без дополнительной проверки. Чтобы убедиться в достоверности даты, выполняется проверка в самом приложении.

Величины года, представленные двумя разрядами, допускают неоднозначное толкование, так как неизвестно столетие. MySQL интерпретирует двухразрядные величины года по следующим правилам:

Величины года в интервале 00-69 преобразуются в 2000-2069.

Величины года в интервале 70-99 преобразуются в 1970-1999.

Тип данных TIME

MySQL извлекает и выводит величины типа TIME в формате 'HH:MM:SS' (или в формате 'HHH:MM:SS' для больших значений часов). Величины TIME могут изменяться в пределах от '-838:59:59' до '838:59:59'. Причина того, что «часовая» часть величины может быть настолько большой, заключается в том, что тип TIME может использоваться не только для представления времени дня (которое должно быть меньше 24 часов), но также для представления общего истекшего времени или временного интервала между двумя событиями (который может быть значительно больше 24 часов или даже отрицательным).

Величины TIME могут быть заданы в различных форматах.

Как строка в формате 'D HH:MM:SS.дробная часть' (следует учитывать, что MySQL пока не обеспечивает хранения дробной части величины в столбце рассматриваемого типа). Можно также использовать одно из следующих «облегченных» представлений: HH:MM:SS.дробная часть, HH:MM:SS, HH:MM, D HH:MM:SS, D HH:MM, D HH или SS. Здесь D - это дни из интервала значений 0-33.

Как строка без разделителей в формате 'HHMMSS', при условии, что строка интерпретируется как дата. Например, величина '101112' понимается как '10:11:12', но величина '109712' будет недопустимой (значение раздела минут является абсурдным) и преобразуется в '00:00:00'.

Как число в формате HHMMSS, при условии, что строка интерпретируется как дата. Например, величина 101112 понимается как '10:11:12'. MySQL понимает и следующие альтернативные форматы: SS, MMSS, HHMMSS, HHMMSS.дробная часть. При этом следует учитывать, что хранения дробной части MySQL пока не обеспечивает.

Как результат выполнения функции, возвращающей величину, приемлемую в контексте типа данных типа TIME (например, такой функции, как CURRENT\_TIME).

Для величин типа TIME, представленных как строки, содержащие разделительные знаки между частями значения времени, нет необходимости указывать два разряда для значений часов, минут или секунд, меньших 10. Так, величина '8:3:2' эквивалентна величине '08:03:02'.

Будьте внимательны в отношении использования «укороченных» величин TIME в столбце типа TIME. MySQL интерпретирует выражения без разделительных двоеточий исходя из предположения, что крайние справа разряды представляют секунды (MySQL интерпретирует величины TIME как общее истекшее время, а не как время дня). Например, можно подразумевать, что величины '1112' и 1112 обозначают '11:12:00' (11 часов и 12 минут дня по показаниям часов), но MySQL понимает их как '00:11:12' (11 минут, 12 секунд). Подобно этому, '12' и 12 интерпретируются как '00:00:12'. Величины TIME с разделительными двоеточиями, наоборот, всегда трактуются как время дня. Т.е. выражение '11:12' будет пониматься как '11:12:00', а не '00:11:12'.

Величины, лежащие вне разрешенного интервала TIME, но во всем остальном представляющие собой допустимые значения, усекаются до соответствующей граничной точки данного интервала. Например, величины '-850:00:00' и '850:00:00' преобразуются соответственно в '-838:59:59' и '838:59:59'.

Недопустимые значения величин TIME преобразуются в значение '00:00:00'. Отметим, что поскольку выражение '00:00:00' само по себе представляет разрешенное значение величины TIME, то по хранящейся в таблице величине '00:00:00' невозможно определить, была ли эта величина изначально задана как '00:00:00' или является преобразованным значением недопустимой величины.

Тип данных YEAR

Тип YEAR - это однобайтный тип данных для представления значений года.

MySQL извлекает и выводит величины YEAR в формате YYYY. Диапазон возможных значений - от 1901 до 2155.

Величины типа YEAR могут быть заданы в различных форматах:

Как четырехзначная строка в интервале значений от '1901' до '2155'.

Как четырехзначное число в интервале значений от 1901 до 2155.

Как двухзначная строка в интервале значений от '00' до '99'. Величины в интервалах от '00' до '69' и от '70' до '99' при этом преобразуются в величины YEAR в интервалах от 2000 до 2069 и от 1970 до 1999 соответственно.

Как двухзначное число в интервале значений от 1 до 99. Величины в интервалах от 1 до 69 и от 70 до 99 при этом преобразуются в величины YEAR в интервалах от 2001 до 2069 и от 1970 до 1999 соответственно. Необходимо принять во внимание, что интервалы для двухзначных чисел и двухзначных строк несколько различаются, так как нельзя указать «ноль» непосредственно как число и интерпретировать его как 2000. Необходимо задать его как строку '0' или '00', или же оно будет интерпретировано как 0000.

Как результат выполнения функции, возвращающей величину, приемлемую в контексте типа данных YEAR (такой как NOW()).

Недопустимые величины YEAR преобразуются в 0000.

**Задание.**

Создайте в MySQL таблицу, содержащую все перечисленные типы данных. Заполните ее не менее 3 записями. Сделайте в ней поиск. Выполните последующие запросы. По возможности в качестве одного из аргументов использовать сегодняшнюю дату (текущее время) плюс НОМЕР СВОЕГО ВАРИАНТА.

Функции даты и времени

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

 SELECT something FROM tbl\_name

 WHERE TO\_DAYS(NOW()) - TO\_DAYS(date\_col) <= 30;

DAYOFWEEK(date)

Возвращает индекс дня недели для аргумента date (1 = воскресенье, 2 = понедельник, ... 7 = суббота). Эти индексные величины соответствуют стандарту ODBC.

SELECT DAYOFWEEK('1998-02-03');

 3

WEEKDAY(date)

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

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

 1

SELECT WEEKDAY('1997-11-05');

 2

DAYOFMONTH(date)

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

 SELECT DAYOFMONTH('1998-02-03');

 3

DAYOFYEAR(date)

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

 SELECT DAYOFYEAR('1998-02-03');

 34

MONTH(date)

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

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

 2

DAYNAME(date)

Возвращает название дня недели для аргумента date:

 SELECT DAYNAME("1998-02-05");

 'Thursday'

MONTHNAME(date)

Возвращает название месяца для аргумента date:

 SELECT MONTHNAME("1998-02-05");

 'February'

QUARTER(date)

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

 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 промежутке 0-53 |
| 1 | Неделя начинается с понедельника; возвращаемое значение - в промежутке 0-53 |
| 2 | Неделя начинается с воскресенья; возвращаемое значение - в промежутке 1-53 |
| 3 | Неделя начинается с понедельника; возвращаемое значение - в промежутке 1-53 (ISO 8601) |

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

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:

SELECT YEAR('2000-01-01'), WEEK('2000-01-01',0);

 2000, 0

 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.

Если нужно уточнить корректную неделю в году, тогда можно использовать 2 или 3 как опциональный аргумент или использовать YEARWEEK()

 SELECT YEARWEEK('2000-01-01');

 199952

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

 52

YEAR(date)

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

 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:

 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:

 SELECT HOUR('10:05:03');

 10

MINUTE(time)

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

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

 5

SECOND(time)

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

 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:

 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:

 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. Функции ADDDATE() и SUBDATE() - синонимы для DATE\_ADD() иDATE\_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 DATEили DATETIME (см. пример ниже). Аргумент date является величиной типа DATETIME 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 типаDATE. В других случаях результат представляет собой величину DATETIME:

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

 1998-01-01 00:00:00

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

 1998-01-01

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

 1997-12-31 23:59:59

 SELECT DATE\_ADD("1997-12-31 23:59:59", INTERVAL 1 SECOND);

 1998-01-01 00:00:00

 SELECT DATE\_ADD("1997-12-31 23:59:59", INTERVAL 1 DAY);

 1998-01-01 23:59:59

 SELECT DATE\_ADD("1997-12-31 23:59:59", INTERVAL "1:1" MINUTE\_SECOND);

 1998-01-01 00:01:00

 SELECT DATE\_SUB("1998-01-01 00:00:00", INTERVAL "1 1:1:1" DAY\_SECOND);

 1997-12-30 22:58:59

 SELECT DATE\_ADD("1998-01-01 00:00:00", INTERVAL "-1 10" DAY\_HOUR);

 1997-12-30 14:00:00

 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:

 SELECT DATE\_ADD("1999-01-01", INTERVAL 1 DAY);

 1999-01-02

 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:

 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.

 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

TO\_DAYS(date)

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

 SELECT TO\_DAYS(950501);

 728779

 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:

 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, число, 4 разряда, используется с '%V' |
| %x | Год для недели, где воскресенье считается первым днем недели, число, 4 разряда, используется с '%v' |
| %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. Используется с '%X' |
| %v | Неделя (01..53), где понедельник считается первым днем недели. Используется с '%x' |
| %% | Литерал '%'. |

Все другие символы просто копируются в результирующее выражение без интерпретации:

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'

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:

 SELECT CURDATE();

 '1997-12-15'

 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:

 SELECT CURTIME();

 '23:50:26'

 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:

 SELECT NOW();

 '1997-12-15 23:50:26'

 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 функцияUNIX\_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 типаDATE, строку DATETIME, величину типа TIMESTAMP или число в формате YYMMDD или YYYYMMDD местного времени:

SELECT UNIX\_TIMESTAMP();

 882226357

 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:

 SELECT FROM\_UNIXTIME(875996580);

 '1997-10-04 22:23:00'

 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 функции DATE\_FORMAT():

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:

 SELECT SEC\_TO\_TIME(2378);

 '00:39:38'

 SELECT SEC\_TO\_TIME(2378) + 0;

 3938

TIME\_TO\_SEC(time)

Возвращает аргумент time, преобразованный в секунды:

 SELECT TIME\_TO\_SEC('22:23:00');

 80580

 SELECT TIME\_TO\_SEC('00:39:38');

 2378

**Контрольные вопросы.**

1. Тип данных YEAR.

2. Тип данных TIME.

3. Типы данных DATETIME, DATE и TIMESTAMP.

4. Конвертирование дат с неоднозначным годом.

5. Какие существуют функции даты и времени?