날짜와 시간 함수

DATE 값을 기대하는 함수들은 일반적으로 DATETIME 값을 수용하고, TIME 부분은 무시한다. TIME 값을 기대하는 함수들은 일반적으로 DATETIME 값을 수용하고, DATE 부분은 무시한다.

현재 날짜나 시간을 반환(return)하는 함수들은 쿼리가 실행될 때 단 한번만 그 값을 구한다. 이것은 한 쿼리 안에 NOW()와 같은 함수가 여러번 사용되었을 경우에도 모두 같은 결과값을 참조한다는 것을 의미한다. 이 원칙은 CURDATE(), CURTIME(), UTC_DATE(), UTC_TIME(), UTC_TIMESTAMP() 등의 함수에도 적용된다.

MySQL 4.1.3 버전부터 제공된 CURRENT_TIMESTAMP(), CURRENT_TIME(), CURRENT_DATE(), FROM_UNIXTIME() 함수들은 연결 상태의 현재 시간대에 해당되는 반환값을 갖는다. 또한 UNIX_TIMESTAMP()도 그 인자(argument)가 현재 시간대에 해당되는 DATETIME 값이라는 것을 가정한다.

다음 함수 설명들의 반환 값 범위는 완전한 날짜를 요구한다. 날짜가 '0'이거나 '2001-11-00'처럼 불완전하다면, DATE 부분을 추출하는 함수는 '0'을 반환할 것이다.
예를 들어, DAYOFMONTH('2001-11-00')은 '0'을 반환한다.

- ADDDATE(date,INTERVAL expr type) , ADDDATE(expr,days)

  ADDDATE()는 두번째 인자에서 INTERVAL과 함께 사용되면 DATE_ADD()의 별칭이 된다. 마찬가지로 SUBDATE()는 DATE_SUB()의 별칭이다. INTERVAL 인자에 관해서는 DATE_ADD() 설명을 참조하라.

  mysql> SELECT DATE_ADD('1998-01-02', INTERVAL 31 DAY); ☞ '1998-02-02'
  mysql> SELECT ADDDATE('1998-01-02', INTERVAL 31 DAY); ☞ '1998-02-02'

  MySQL 4.1.1 버전부터 두번째 문법이 허용되었다. expr 가 DATE 또는 DATETIME 형식일 때, days 는 expr 에 추가되는 일수이다.

  mysql> SELECT ADDDATE('1998-01-02', 31); ☞ '1998-02-02'

- ADDTIME(expr,expr2)

  ADDTIME()는 expr 에 expr2 를 더하고 그 결과를 반환한다. expr 는 TIME 또는 DATETIME 형식이고, expr2 는 시간 표현이다.

  mysql> SELECT ADDTIME('1997-12-31 23:59:59.999999', '1 1:1:1.000002');
  ☞ '1998-01-02 01:01:01.000001'
  mysql> SELECT ADDTIME('01:00:00.999999', '02:00:00.999998');
  ☞ '03:00:01.999997'
  ADDTIME()는 MySQL 4.1.1 버전에서 추가되었다.

- CONVERT_TZ(dt,from_tz,to_tz)

  CONVERT_TZ()는 DATETIME 값 dt 를 from_tz 시간대에서 to_tz 시간대로 변환하고, 결과값을 반환한다. 이 함수는 인자가 유효하지 않으면 NULL 값을 반환한다.

  from_tz 에서 UTC으로 변환될 때 입력값이 TIMESTAMP 형의 범위를 벗어나면 변환은 일어나지 않는다.

  mysql> SELECT CONVERT_TZ('2004-01-01 12:00:00','GMT','MET');
  ☞ '2004-01-01 13:00:00'
  mysql> SELECT CONVERT_TZ('2004-01-01 12:00:00','+00:00','-07:00');
  ☞ '2004-01-01 05:00:00'
  'MET'이나 'Europe/Moscow'와 같은 시간대를 사용하기 위해서는, 시간대 표가 적절하게 설정되어야 한다.

  CONVERT_TZ()는 MySQL 4.1.3 버전에서 추가되었다.

- CURDATE()

  함수가 문자열이나 숫자로 사용되었는지 문맥에 따라서 'YYYY-MM-DD'이나 YYYYMMDD 형식으로 현재 날짜를 반환한다.

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

- CURRENT_DATE , CURRENT_DATE()

  CURRENT_DATE와 CURRENT_DATE()는 CURDATE()의 별칭이다.

- CURTIME()

  함수가 문자열이나 숫자로 사용되었는지 문맥에 따라서 'HH:MM:SS'이나 HHMMSS 형식으로 현재 시간을 반환한다.

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

- CURRENT_TIME , CURRENT_TIME()

  CURRENT_TIME와 CURRENT_TIME()는 CURTIME()의 별칭이다.

- CURRENT_TIMESTAMP , CURRENT_TIMESTAMP()

  CURRENT_TIMESTAMP와 CURRENT_TIMESTAMP()는 NOW()의 별칭이다.

- DATE(expr)

  날짜(date)나 DATETIME 표현 expr 에서 DATE 부분을 추출한다.

mysql> SELECT DATE('2003-12-31 01:02:03'); ☞ '2003-12-31'
DATE()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- DATEDIFF(expr,expr2)

  DATEDIFF()는 시작 날짜 expr 와 마지막 날짜 expr2 사이의 일수를 반환한다. expr 와 expr2 는 날짜(date) 또는 date-and-time 표현이다. 반환값의 DATE 부분만 계산된다.

  mysql> SELECT DATEDIFF('1997-12-31 23:59:59','1997-12-30'); ☞ 1
  mysql> SELECT DATEDIFF('1997-11-30 23:59:59','1997-12-31'); ☞ -31
  DATEDIFF()는 MySQL 4.1.1 버전에서 추가되었다.

- DATE_ADD(date,INTERVAL expr type) , DATE_SUB(date,INTERVAL expr type)

  이 함수들은 날짜 계산을 수행한다. date 는 시작 날짜를 지정하는 DATETIME 또는 DATE 값이다. expr 는 시작 날짜로부터 더하거나 뺀 간격 값을 지정하는 표현이다. expr 는 문자열이다. 마이너스('-')로 시작될 수도 있다. type 는 어떻게 해석할지를 지정하는 키워드이다.

  INTERVAL 키워드와 type 지정자는 대소문자를 구분하지 않는다.

  다음 표는 type 와 expr 인자가 어떤 관계인지 보여준다.

type 값	기대되는 expr 형식
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.MICROSECONDS'
MINUTE_SECOND	'MINUTES:SECONDS'
HOUR_MICROSECOND	'HOURS.MICROSECONDS'
HOUR_SECOND	'HOURS:MINUTES:SECONDS'
HOUR_MINUTE	'HOURS:MINUTES'
DAY_MICROSECOND	'DAYS.MICROSECONDS'

DAY_SECOND	'DAYS HOURS:MINUTES:SECONDS'
DAY_MINUTE	'DAYS HOURS:MINUTES'
DAY_HOUR	'DAYS HOURS'
YEAR_MONTH	'YEARS-MONTHS'

MySQL은 expr 형식 안에서 어떤 구문 구획자도 허용한다. 표에서 볼 수 있는 것들은 제안된 구획자들이다. date 인자가 DATE 값이고 단지 YEAR, MONTH, DAY 만 계산하고자 한다면(TIME 부분이 필요 없다면), 결과는 DATE 값이다. 그렇지 않다면, 결과는 DATETIME 값이다.

MySQL 3.23 버전부터, INTERVAL expr type 는 다른 부분이 DATE 또는 DATETIME 값으로 표현되어 있다면 어느 한쪽이라도 + 연산자의 사용을 허용한다. - 연산자는 오른쪽에만 허용된다. 간격에서 DATE 또는 DATETIME 값을 빼는 것은 무의미하기 때문이다. (아래 예문을 참조하라.)

```
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'
mysql> SELECT DATE_ADD('1992-12-31 23:59:59.000002', INTERVAL '1.999999'
SECOND_MICROSECOND);
        ☞ '1993-01-01 00:00:01.000001'
```

지정한 간격(interval) 값이 너무 짧다면(type 키워드로부터 기대되는 모든 간격 부분이 포함되어 있지 않다면), MySQL은 간격 값의 왼쪽 부분을 남겼다고 가정한다. 예를 들어, type DAY_SECOND를 지정했다면, expr 값은 일, 시, 분, 초 부분이 기대된다. '1:10'과 같은 값을 지정했다면, MySQL은 일, 시 부분이 없는 분, 초 값이라고 가정한다. 다르게 말하면, '1:10' DAY_SECOND는 '1:10' MINUTE_SECOND과 동일한 값으로 해석된다. 이것은 MySQL가 TIME 값을 시각보다 시간으로 해석하는 것과 비슷하다.

TIME 부분을 포함하는 어떤 값에서 date 를 더하거나 뺀다면, 결과는 자동으로 DATETIME 값으로 변환된다.

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'
기형적인 날짜를 입력하면 결과는 NULL이 된다. 만일 MONTH, YEAR_MONTH, 또는 YEAR를 더해서 새로운 달의 일수보다 더 큰 날짜가 된다면, 날짜는 새로운 달의 마지막 날로 보정된다.

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

- DATE_FORMAT(date,format)

format 문자열에 따라 date 값을 형식화한다. format 문자열에는 다음 지정자들이 사용된다.

지정자	설명
%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
%u	Week (00..53), where Monday is the first day of the week
%V	Week (01..53), where Sunday is the first day of the week; used with %X
%v	Week (01..53), where Monday is the first day of the week; 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 '%'.

다른 모든 문자들은 변환 없이 결과에 그대로 복사된다.

%v, %V, %x, %X format 지정자들은 MySQL 3.23.8 버전부터 사용이 가능하고, %f는 MySQL 4.1.1 버전부터 가능하다.

MySQL 3.23 버전부터는 '%' 문자가 format 지정문자 앞에 요구된다. 그 이전 버번에서 '%'는 선택사양이다.

월일의 범위가 '0'으로 시작되기 때문에 MySQL 3.23 버전부터는 '2004-00-00'와 같은 불완전한 날짜가 허용된다.

```
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'
```

- DAY(date)

  DAY()는 DAYOFMONTH()의 별칭이다. MySQL 4.1.1 버전부터 사용이 가능하다.

- DAYNAME(date)
  date 에 대한 요일 이름을 반환한다.

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

- DAYOFMONTH(date)
  date 에 대한 당월의 날짜를 반환한다. (범위 1~31)

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

- DAYOFWEEK(date)
  date 에 대하여 요일 색인(1 = 일요일, 2 = 월요일, ..., 7 = 토요일)을 반환한다. 이 색인값들은 ODBC 표준에 따른다.

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

- DAYOFYEAR(date)
  date 가 해당 연도에 몇일째인지 반환한다. (범위 1~366)

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

- EXTRACT(type FROM date)
  EXTRACT() 함수는 DATE_ADD()나 DATE_SUB()와 같은 종류의 간격 지정자를 사용하지만, 날짜를 계산하는 게 아니라 날짜로부터 부분을 추출한다.

  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
  mysql> SELECT EXTRACT(MICROSECOND FROM '2003-01-02 10:30:00.00123'); ☞ 123

  EXTRACT()는 MySQL 3.23.0 버전에서 추가되었다.

- FROM_DAYS(N)
  일수 N 가 주어지면, DATE 값을 반환한다.

  mysql> SELECT FROM_DAYS(729669); ☞ '1997-10-07'
  FROM_DAYS()는 그레고리안 달력의 출현(1582년) 이전의 값을 사용할 수 있도록 계획되지 않았다. 달력이 바뀌었을 때 손실된 날짜는 고려하지 않는다.

- FROM_UNIXTIME(unix_timestamp) , FROM_UNIXTIME(unix_timestamp,format)
  함수에 사용된 문맥이 문자열인지 숫자인지에 따라 'YYYY-MM-DD HH:MM:SS' 또는 YYYYMMDDHHMMSS format 값으로 unix_timestamp 인자가 표시되어 반환된다.

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

format 이 주어진다면 결과는 format 문자열에 따라 형식화된다. format 은 DATE_FORMAT() 함수에 쓰이는 지정자를 똑같이 사용한다.

mysql> SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(), '%Y %D %M %h:%i:%s %x');
☞ '2003 6th August 06:22:58 2003'

- GET_FORMAT(DATE|TIME|DATETIME, 'EUR'|'USA'|'JIS'|'ISO'|'INTERNAL')
  형식 문자열을 반환한다. 이 함수는 DATE_FORMAT()과 STR_TO_DATE() 함수와 조합할 때 유용하다.

  첫번째 인자로는 3가지 가능한 값이 있고, 두번째 인자로는 5가지 가능한 값이 있어서 결과적으로 15가지 형식 문자열이 가능하다. (사용되는 지정자를 위해서 DATE_FORMAT() 설명을 참조하라.)

함수 호출	결과
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'

ISO 형식은 ISO 8601이 아니라 ISO 9075이다.

MySQL 4.1.4 버전부터는 TIMESTAMP 또한 사용할 수 있게 되었다. GET_FORMAT() 함수는 같은 값을 DATETIME 형식으로 반환한다.

mysql> SELECT DATE_FORMAT('2003-10-03',GET_FORMAT(DATE,'EUR')); ☞ '03.10.2003'
mysql> SELECT STR_TO_DATE('10.31.2003',GET_FORMAT(DATE,'USA')); ☞ 2003-10-31
GET_FORMAT()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- HOUR(time)

  time 에서 시간을 반환한다. (범위 0~23)

  mysql> SELECT HOUR('10:05:03'); ☞ 10
  그러나 TIME 값의 크기는 실제로 훨씬 더 크다. HOUR는 23보다 더 큰 값을 반환할 수 있다.

  mysql> SELECT HOUR('272:59:59'); ☞ 272

- LAST_DAY(date)

  DATE 또는 DATETIME 값을 입력하면 당월의 마지막 날에 대한 상응하는 값을 반환한다. 인자가 유효하지 않으면 NULL을 반환한다.

  mysql> SELECT LAST_DAY('2003-02-05'); ☞ '2003-02-28'
  mysql> SELECT LAST_DAY('2004-02-05'); ☞ '2004-02-29'
  mysql> SELECT LAST_DAY('2004-01-01 01:01:01'); ☞ '2004-01-31'
  mysql> SELECT LAST_DAY('2003-03-32'); ☞ NULL
  LAST_DAY()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- LOCALTIME , LOCALTIME()

  LOCALTIME과 LOCALTIME()은 NOW()의 별칭이다.

  두 함수는 MySQL 4.0.6 버전에서 추가되었다.

- LOCALTIMESTAMP , LOCALTIMESTAMP()

  LOCALTIMESTAMP와 LOCALTIMESTAMP()는 NOW()의 별칭이다.

  두 함수는 MySQL 4.0.6 버전에서 추가되었다.

- MAKEDATE(year,dayofyear)

  year 와 dayofyear 값이 주어지면 날짜를 반환한다. dayofyear 는 0보다 커야 한다. 그렇지 않으면 결과는 NULL이다.

  mysql> SELECT MAKEDATE(2001,31), MAKEDATE(2001,32); ☞ '2001-01-31', '2001-02-01'
  mysql> SELECT MAKEDATE(2001,365), MAKEDATE(2004,365); ☞ '2001-12-31', '2004-12-30'
  mysql> SELECT MAKEDATE(2001,0); ☞ NULL
  MAKEDATE()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- MAKETIME(hour,minute,second)
  hour,minute,second 인자로부터 계산된 시간 값을 반환한다.

  mysql> SELECT MAKETIME(12,15,30); ☞ '12:15:30'
  MAKETIME()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- MICROSECOND(expr)
  TIME 또는 DATETIME 형식의 expr 로부터 마이크로초를 반환한다. (범위 0~999999)

  mysql> SELECT MICROSECOND('12:00:00.123456'); ☞ 123456
  mysql> SELECT MICROSECOND('1997-12-31 23:59:59.000010'); ☞ 10
  MICROSECOND()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- MINUTE(time)
  time 에 대하여 몇 분인지 반환한다. (범위 0~59)

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

- MONTH(date)
  date 에 대하여 몇 월인지 반환한다. (범위 1~12)

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

- MONTHNAME(date)
  date 에 대하여 당월의 영문 이름을 반환한다.

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

- NOW()
  함수에 사용된 문맥에 따라 'YYYY-MM-DD HH:MM:SS' 또는 YYYYMMDDHHMMSS 형식으로 현재 날짜와 시간을 반환한다.

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

- PERIOD_ADD(P,N)
  기간 P 에 N 월을 더한다(YYMM 또는 YYYYMM 형식으로). YYYYMM 형식으로 결과를 반환한다. 기간 P 가 DATE 값이 아니라는 것에 주의하라.

  mysql> SELECT PERIOD_ADD(9801,2); ☞ 199803

- PERIOD_DIFF(P1,P2)

기간 P1, P2 사이의 개월수를 반환한다. P1 과 P2 는 YYMM 또는 YYYYMM 형식이어야 한다. 기간 P1, P2 가 DATE 값이 아니라는 것에 주의하라.

mysql> SELECT PERIOD_DIFF(9802,199703); ☞ 11

- QUARTER(date)

  date 가 몇 분기인지 반환한다. (범위 1~4)

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

- SECOND(time)

  time 에서 초 값을 반환한다. (범위 0~59)

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

- SEC_TO_TIME(seconds)

  함수가 어떤 문맥으로 사용되었는지에 따라 seconds 인자를 'HH:MM:SS' 또는 HHMMSS 형식으로 변환시켜서 반환한다.

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

- STR_TO_DATE(str,format)

  이 함수는 DATE_FORMAT() 함수의 역기능이다. 문자열 str 와 형식 문자열 format 을 입력받는다. STR_TO_DATE()는 형식 문자열이 날짜와 시간을 모두 포함하고 있다면 DATETIME 값을 반환한다. 그렇지 않고 날짜나 시간 둘 중 한 부분만을 포함한다면 DATE 또는 TIME 값을 반환한다.

  str 에 포함된 DATE, TIME 또는 DATETIME 값은 format 에 의해 지정된 형식으로 주어져야 한다. format 에 사용할 수 있는 지정자에 대해서는 DATE_FORMAT() 설명을 참조하라. 다른 모든 문자는 해석되지 않고 그대로 반영된다. 만일 str 가 유효하지 않은 값을 포함한다면 NULL이 반환된다. MySQL 5.0.3 버전부터는 잘못된 값 또한 경고를 발생한다.

  mysql> SELECT STR_TO_DATE('03.10.2003 09.20', '%d.%m.%Y %H.%i');
  ☞ '2003-10-03 09:20:00'
  mysql> SELECT STR_TO_DATE('10arp', '%carp');
  ☞ '0000-10-00 00:00:00'
  mysql> SELECT STR_TO_DATE('2003-15-10 00:00:00', '%Y-%m-%d %H:%i:%s');
  ☞ NULL

  어떤 달의 일수보다 큰 일수를 가진 날짜는 1-31 범위 안에서 허용된다. 또한 '0'이나 '0'값을 가진 날짜도 허용된다.

  mysql> SELECT STR_TO_DATE('00/00/0000', '%m/%d/%Y'); ☞ '0000-00-00'
  mysql> SELECT STR_TO_DATE('04/31/2004', '%m/%d/%Y'); ☞ '2004-04-31'

STR_TO_DATE()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- SUBDATE(date,INTERVAL expr type) , SUBDATE(expr,days)

  두번째 인자 INTERVAL 형식을 포함하여 사용되었을 때 SUBDATE()는 DATE_SUB()의 별칭이다. INTERVAL 인자에 대한 정보는 DATE_ADD() 설명을 참조하라.

  mysql> SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY); ☞ '1997-12-02'
  mysql> SELECT SUBDATE('1998-01-02', INTERVAL 31 DAY); ☞ '1997-12-02'
  MySQL 4.1.1 버전부터 두번째 문법이 허용된다. expr 는 DATE 또는 DATETIME 형식이고 days 는 expr 에서 뺄 일수이다.

  mysql> SELECT SUBDATE('1998-01-02 12:00:00', 31); ☞ '1997-12-02 12:00:00'

- SUBTIME(expr,expr2)

  SUBTIME()는 expr 에서 expr2 를 빼고 그 값을 반환한다. expr 는 TIME 또는 DATETIME 형식이고, expr2 는 TIME 형식이다.

  mysql> SELECT SUBTIME('1997-12-31 23:59:59.999999', '1 1:1:1.000002');
  ☞ '1997-12-30 22:58:58.999997'
  mysql> SELECT SUBTIME('01:00:00.999999', '02:00:00.999998');
  ☞ '-00:59:59.999999'
  SUBTIME()는 MySQL 4.1.1 버전에서 추가되었다.

- SYSDATE()

  SYSDATE()는 NOW()의 별칭이다.

- TIME(expr)

  TIME 또는 DATETIME 형식의 expr 에서 TIME 부분을 추출한다.

  mysql> SELECT TIME('2003-12-31 01:02:03'); ☞ '01:02:03'
  mysql> SELECT TIME('2003-12-31 01:02:03.000123'); ☞ '01:02:03.000123'
  TIME()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- TIMEDIFF(expr,expr2)

  TIMEDIFF()는 시작 시간 expr 와 마지막 시간 expr2 와의 차이를 TIME 값으로 반환한다. expr 와 expr2 는 TIME 또는 DATETIME 형식이고, 두 형식은 같아야 한다.

  mysql> SELECT TIMEDIFF('2000:01:01 00:00:00', '2000:01:01 00:00:00.000001');
  ☞ '-00:00:00.000001'
  mysql> SELECT TIMEDIFF('1997-12-31 23:59:59.000001', '1997-12-30 01:01:01.000002');
  ☞ '46:58:57.999999'
  TIMEDIFF()는 MySQL 4.1.1 버전에서 추가되었다.

- TIMESTAMP(expr) , TIMESTAMP(expr,expr2)
  인자 하나만을 사용한다면, DATE 또는 DATETIME expr 를 DATETIME 값으로 반환한다. 인자 두 개를 사용한다면, DATE 또는 DATETIME 형식의 expr 에 TIME 형식의 expr2 를 더하고 그 DATETIME 값을 반환한다.

  mysql> SELECT TIMESTAMP('2003-12-31'); ☞ '2003-12-31 00:00:00'
  mysql> SELECT TIMESTAMP('2003-12-31 12:00:00','12:00:00'); ☞ '2004-01-01 00:00:00'
  TIMESTAMP()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- TIMESTAMPADD(interval,int_expr,DATETIME_expr)
  DATE 또는 DATETIME 형식의 DATETIME_expr 에 정수 형식의 int_expr 를 더한다. int_expr 의 단위는 interval 인자로 주어지는데, 다음 값 가운데 하나이어야 한다. FRAC_SECOND, SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR.

  interval 값은 상기된 키워드 가운데 하나를 지정하거나 SQL_TSI_ 접두사를 사용할 수 있다. 예를 들어, DAY 또는 SQL_TSI_DAY 둘 다 모두 허용된다.

  mysql> SELECT TIMESTAMPADD(MINUTE,1,'2003-01-02'); ☞ '2003-01-02 00:01:00'
  mysql> SELECT TIMESTAMPADD(WEEK,1,'2003-01-02'); ☞ '2003-01-09'
  TIMESTAMPADD()는 MySQL 5.0.0 버전부터 사용이 가능하다.

- TIMESTAMPDIFF(interval,DATETIME_expr1,DATETIME_expr2)
  DATE 또는 DATETIME 형식의 DATETIME_expr1,DATETIME_expr2 사이의 격차를 정수값으로 반환한다. 결과값의 단위는 interval 인자에 의해 주어진다. interval 의 허용값은 TIMESTAMPADD() 함수와 같다.

  mysql> SELECT TIMESTAMPDIFF(MONTH,'2003-02-01','2003-05-01'); ☞ 3
  mysql> SELECT TIMESTAMPDIFF(YEAR,'2002-05-01','2001-01-01'); ☞ -1
  TIMESTAMPDIFF()는 MySQL 5.0.0 버전부터 사용이 가능하다.

- TIME_FORMAT(time,format)
  이 함수는 DATE_FORMAT() 함수처럼 사용되지만, format 문자열은 시, 분, 초에 관련된 지정자만을 포함할 수 있다. 다른 지정자들은 NULL 값이나 '0'을 발생한다.

  time 값이 TIME 부분에서 23보다 큰 값을 갖는다면, %H와 %k 시간 지정자는 일상적인 범위 0-23보다 더 큰 값을 발생한다. 다른 시간 지정자들은 12 법(modulo)의 값을 발생한다.

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

- TIME_TO_SEC(time)
  time 인자를 초로 변환하여 반환한다.

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

- TO_DAYS(date)

날짜 date 가 주어지면, 일수를 반환한다. (0년부터의 일수).

mysql> SELECT TO_DAYS(950501); ☞ 728779
mysql> SELECT TO_DAYS('1997-10-07'); ☞ 729669
TO_DAYS()는 그레고리안 달력의 출현(1582년) 이전의 값을 사용할 수 있도록 계획되지 않았다. 달력이 바뀌었을 때 손실된 날짜는 고려하지 않는다.

MySQL는 날짜에 있는 2자리 형식의 연도를 4자리 형식으로 변환한다는 것을 기억하라. 예를 들어, '1997-10-07'과 '97-10-07'는 동일한 날짜로 간주한다.

mysql> SELECT TO_DAYS('1997-10-07'), TO_DAYS('97-10-07'); ☞ 729669, 729669
1582년 이전의 다른 날짜에 대해서는 이 함수는 결과값이 정의되지 않았다.

- UNIX_TIMESTAMP() , UNIX_TIMESTAMP(date)

인자 없이 호출이 된다면, 부호없는 정수의 유닉스 시간('1970-01-01 00:00:00' GMT부터 계산된 초)을 반환한다. UNIX_TIMESTAMP()가 date 인자와 함께 호출된다면, '1970-01-01 00:00:00' GMT부터 계산된 초 값을 반한다. date 는 DATE 문자열, DATETIME 문자열, TIMESTAMP 문자열, YYMMDD 또는 YYYYMMDD 형식의 숫자를 허용한다.

mysql> SELECT UNIX_TIMESTAMP(); ☞ 882226357
mysql> SELECT UNIX_TIMESTAMP('1997-10-04 22:23:00'); ☞ 875996580
UNIX_TIMESTAMP가 TIMESTAMP 형식으로 사용되었을 때 이 함수는 내부의 TIMESTAMP 값을 직접 반환한다. 무조건 문자열을 유닉스 시간으로 변환하지 않는다. UNIX_TIMESTAMP()에 범위에서 벗어난 날짜를 입력했다면 0이 반환되지만, 기본적인 범위만 확인된다는 것에 주의하라. (연도는 1970-2037, 월 01-12, 일 01-31)

- UTC_DATE , UTC_DATE()

함수가 사용된 문맥에 따라 'YYYY-MM-DD' 또는 YYYYMMDD 형식으로 현재의 UTC 날짜 값을 반환한다.

mysql> SELECT UTC_DATE(), UTC_DATE() + 0; ☞ '2003-08-14', 20030814
UTC_DATE()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- UTC_TIME , UTC_TIME()

함수가 사용된 문맥에 따라 'HH:MM:SS' 또는 HHMMSS 형식으로 현재의 UTC 시간 값을 반환한다.

mysql> SELECT UTC_TIME(), UTC_TIME() + 0; ☞ '18:07:53', 180753

UTC_TIME()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- **UTC_TIMESTAMP , UTC_TIMESTAMP()**

  함수가 사용된 문맥에 따라 'YYYY-MM-DD HH:MM:SS' 또는 YYYYMMDDHHMMSS 형식으로 현재의 UTC 일시 값을 반환한다.

  mysql> SELECT UTC_TIMESTAMP(), UTC_TIMESTAMP() + 0;
  ☞ '2003-08-14 18:08:04', 20030814180804
  UTC_TIMESTAMP()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- **WEEK(date[,mode])**

  date 에 대하여 몇번째 주인지 반환한다. 2개 인자를 사용하는 형식에서는, 한 주의 시작을 일요일로 할 것인지 월요일로 할 것인지, 결과값의 범위를 0-53으로 할 것인지 1-53으로 할 것인지를 지정할 수 있다. mode 인자가 생략되면 시스템 기본값이 사용된다. (MySQL 4.0.14 버전 이전에는 0)

  mode 인자는 아래 표와 같이 작동한다.

모드	한 주의 시작요일	범위
0	Sunday	0-53
1	Monday	0-53
2	Sunday	1-53
3	Monday	1-53
4	Sunday	0-53
5	Monday	0-53
6	Sunday	1-53
7	Monday	1-53

  mode 3은 MySQL 4.0.5 버전부터 사용할 수 있으며, 4 이상의 mode 는 MySQL 4.0.17 버전부터 사용할 수 있다.

  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
  주의 : MySQL 4.0 버전에서 WEEK(date,0)는 미국 달력에 알맞게 변했다. 그 전에 WEEK()는 미국 날짜에서 잘못 계산되었다. (사실상 WEEK(date)와 WEEK(date,0)는 모든 경우에 오류가 있었다.)

  이전 연도의 마지막주에서 날짜가 맞아떨어지면, mode 인자를 2, 3, 6, 7 으로 선택하지 않는 한 MySQL은 0을 반환한다.

  mysql> SELECT YEAR('2000-01-01'), WEEK('2000-01-01',0); ☞ 2000, 0
  어떤 이는 실제로 주어진 날짜가 1999년의 52째주이기 때문에 WEEK() 함수가 52를 반환해야 한다고

주장한다. 우리는 주어진 연도에서 몇 째주인지 반환할 것을 원했기 때문에 그 대신 0을 반환하기로 했다. 이것은 WEEK() 함수를 날짜에서 DATE 부분을 추출하는 다른 함수들과 결합하여 사용할 때 유용하다.

주어진 날짜의 주간 첫날을 포함한 해를 고려한 결과값을 원한다면, mode 인자를 0, 2, 5, 7 로 선택해야 한다.

mysql> SELECT WEEK('2000-01-01',2); ☞ 52
YEARWEEK() 함수를 사용하는 것도 대안이 될 수 있다.

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

- WEEKDAY(date)
  date 에 대한 요일 색인값(0 = 월요일, 1 = 화요일, ... 6 = 일요일)을 반환한다.

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

- WEEKOFYEAR(date)
  date 의 달력에서 몇 주째인지 1-53 범위의 값을 반환한다. 이 함수는 WEEK(date,3)과 동일하다.

  mysql> SELECT WEEKOFYEAR('1998-02-20'); ☞ 8
  WEEKOFYEAR()는 MySQL 4.1.1 버전부터 사용이 가능하다.

- YEAR(date)
  date 에서 해당 연도를 반환한다. (범위 1000-9999)

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

- YEARWEEK(date) , YEARWEEK(date,start)
  date 에 대하여 연도와 몇 주째인지 반환한다. 첫번째 인자는 WEEK()의 첫번째 인자와 정확하게 작동한다. 결과 안의 연도는 첫 주와 마지막 주에 한하여 date 인자의 연도와 다를 수 있다.

  mysql> SELECT YEARWEEK('1987-01-01'); ☞ 198653
  주수는 선택 인자 0 또는 1 에 대하여 WEEK() 함수가 반환하는 것과 다르다. WEEK()는 주어진 연도의 문맥에서 주 값을 반환한다.

  YEARWEEK()는 MySQL 3.23.8 버전에서 추가되었다.