

MySQL

Bekerja dengan Tanggal dan Waktu

Oleh :

Janner Simarmata
sijanner@yahoo.com
<http://simarmata.cogia.net>

*Dipublikasikan dan didedikasikan
untuk perkembangan pendidikan di Indonesia melalui*

MateriKuliah.Com

Lisensi Pemakaian Artikel:

*Seluruh artikel di **MateriKuliah.Com** dapat digunakan, dimodifikasi dan disebarluaskan secara bebas untuk tujuan bukan komersial (nonprofit), dengan syarat tidak menghapus atau merubah atribut Penulis. Hak Atas Kekayaan Intelektual setiap artikel di **MateriKuliah.Com** adalah milik Penulis masing-masing, dan mereka bersedia membagikan karya mereka semata-mata untuk perkembangan pendidikan di Indonesia. **MateriKuliah.Com** sangat berterima kasih untuk setiap artikel yang sudah Penulis kirimkan.*

1.1 Format Tanggal dan Waktu pada MySQL

MySQL menyediakan tipe data DATE dan TIME untuk menyatakan nilai tanggal dan waktu secara terpisah, dan tipe DATETIME untuk kombinasi tanggal dan waktu. Tipe-tipe data ini menggunakan format sebagai berikut :

- Tipe DATE menggunakan format CCYY-MM-DD, dimana CC, YY, MM dan DD mewakili abad, tahun dalam abad itu, bulan dan hari pada bulan itu.
- Tipe TIME menggunakan format hh:mm:ss, dimana hh, mm dan ss adalah jam, menit dan detik.
- Tipe DATETIME menggunakan format CCYY-MM-DD hh:mm:ss.
- Tipe TIMESTAMP juga menggunakan format seperti pada DATETIME yaitu CCYYMMDDhhmmss, tetapi datanya dibuat dalam bentuk string.

```
mysql> desc timestamp_val;
```

Field	Type	Null	Key	Default	Extra
ts	timestamp(14)	YES		NULL	

```
mysql> desc datetime_val;
```

Field	Type	Null	Key	Default	Extra
dt	datetime	YES		NULL	

```
mysql> desc date_val;
```

Field	Type	Null	Key	Default	Extra
d	date	YES		NULL	

```
mysql> desc time_val;
```

Field	Type	Null	Key	Default	Extra
t1	time	YES		NULL	
t2	time	YES		NULL	

```
mysql> SELECT t1, t2 FROM time_val;
```

t1	t2
15:00:00	15:00:00
05:01:30	02:30:20
12:30:20	17:30:45

```
mysql> SELECT d FROM date_val;
```

d
1864-02-28
1900-01-15
1987-03-05

```
| 1999-12-31 |
| 2000-06-04 |
+-----+
```

```
mysql> SELECT dt FROM datetime_val;
```

```
+-----+
| dt                |
+-----+
| 1970-01-01 00:00:00 |
| 1987-03-05 12:30:15 |
| 1999-12-31 09:00:00 |
| 2000-06-04 15:45:30 |
+-----+
```

```
mysql> SELECT ts FROM timestamp_val;
```

```
+-----+
| ts                |
+-----+
| 19700101000000    |
| 19870305123015    |
| 19991231090000    |
| 20000604154530    |
+-----+
```

1.2 Merubah Format Tanggal pada MySQL

Format data tanggal pada MySQL mengikuti format standard dari ISO. Untuk tujuan input data, format yang tidak sama dengan format tersebut tidak akan diterima oleh MySQL. Akan tetapi format yang mendekati format tersebut masih bisa diterima oleh MySQL. Misalnya adalah nilai string 87-1-7 dan 1987-1-7 atau bilangan seperti 870107 dan 19870107. Kedua format itu akan diinterpretasikan oleh MySQL sebagai 1987-01-07 ketika dimasukkan dalam kolom bertipe DATE.

Sedang untuk tujuan menampilkan data, format data yang non-ISO dapat diterima oleh MySQL. Hal ini dapat dilakukan dengan menggunakan fungsi DATE_FORMAT(). Fungsi lain seperti YEAR() dapat juga digunakan untuk mengambil bagian tahun saja dari suatu tanggal.

1.3 Merubah tampilan Tanggal dan Waktu

Fungsi DATE_FORMAT() mempunyai dua argumen yaitu: nilai DATE, DATETIME atau TIMESTAMP dan string yang menyatakan bagaimana menampilkan tanggal dan waktu dengan format tertentu. Sebagai contoh, dapat digunakan %c untuk menampilkan bagian dari tanggal, %Y, %M dan %d untuk menampilkan tahun dalam empat angka, nama bulan, dan tanggal dalam dua angka. Query berikut ini menunjukkan beberapa pemakaian dari fungsi DATE_FORMAT().

```
mysql> SELECT d, DATE_FORMAT(d,'%M %d, %Y') FROM date_val;
```

```
+-----+-----+
| d                | DATE_FORMAT(d,'%M %d, %Y') |
+-----+-----+
| 1864-02-28       | February 28, 1864          |
| 1900-01-15       | January 15, 1900           |
| 1987-03-05       | March 05, 1987             |
| 1999-12-31       | December 31, 1999          |
| 2000-06-04       | June 04, 2000              |
+-----+-----+
```

Bisa juga kita menggunakan alias untuk mengganti judul suatu kolom dengan as :

```
mysql> SELECT d, DATE_FORMAT(d,'%M %d, %Y') AS date FROM date_val;
```

d	date
1864-02-28	February 28, 1864
1900-01-15	January 15, 1900
1987-03-05	March 05, 1987
1999-12-31	December 31, 1999
2000-06-04	June 04, 2000

Daftar format waktu selengkapnya disajikan dibawah ini :

Format	Arti
%Y	Four-digit year
%y	Two-digit year
%M	Complete month name
%b	Month name, initial three letters
%m	Two-digit month of year (01..12)
%c	Month of year (1..12)
%d	Two-digit day of month (01..31)
%e	Day of month (1..31)
%r	12-hour time with AM or PM suffix
%T	24-hour time
%H	Two-digit hour
%i	Two-digit minute
%s	Two-digit second
%%	Literal %

Daftar diatas hanya berguna apabila parameter pada DATE_FORMAT diisi dengan nilai yang punya bagian baik tanggal maupun waktu (*DATETIME* atau *TIMESTAMP*).

Query berikut ini menunjukkan bagaimana menggunakan fungsi DATE_FORMAT dengan nilai yang bertipe data DATETIME.

```
mysql> SELECT dt, DATE_FORMAT(dt,'%c/%e/%y %r') AS format1,  
-> DATE_FORMAT(dt,'%M %e, %Y %T') AS format2 FROM datetime_val;
```

dt	format1	format2
1970-01-01 00:00:00	1/1/70 12:00:00 AM	January 1, 1970 00:00:00
1987-03-05 12:30:15	3/5/87 12:30:15 PM	March 5, 1987 12:30:15
1999-12-31 09:00:00	12/31/99 09:00:00 AM	December 31, 1999 09:00:00
2000-06-04 15:45:30	6/4/00 03:45:30 PM	June 4, 2000 15:45:30

TIME_FORMAT() mirip dengan DATE_FORMAT(), tapi hanya memproses data yang berhubungan dengan waktu dalam format string. TIME_FORMAT() bisa mengolah data dengan tipe TIME, DATETIME atau TIMESTAMP.

```
mysql> SELECT dt, TIME_FORMAT(dt, '%r') AS '12-hour time',  
-> TIME_FORMAT(dt, '%T') AS '24-hour time' FROM datetime_val;
```

dt	12-hour time	24-hour time
----	--------------	--------------

1970-01-01 00:00:00	12:00:00 AM	00:00:00	
1987-03-05 12:30:15	12:30:15 PM	12:30:15	
1999-12-31 09:00:00	09:00:00 AM	09:00:00	
2000-06-04 15:45:30	03:45:30 PM	15:45:30	
+-----+-----+-----+			

1.4 Menentukan tanggal atau waktu sekarang

Beberapa aplikasi membutuhkan tanggal atau waktu sekarang. Informasi semacam ini juga berguna untuk penghitungan tanggal, seperti mencari hari pertama atau terakhir dari suatu bulan atau menentukan tanggal dari hari Rabu minggu depan.

Tanggal atau waktu sekarang tersedia melalui tiga fungsi yaitu fungsi NOW() yang mengembalikan baik tanggal maupun waktu sekarang, serta fungsi CURDATE() dan CURTIME() yang mengembalikan tanggal dan waktu sekarang secara terpisah.

```
mysql> select now(), curdate(),curtime();
```

now()	curdate()	curtime()
2004-10-15 14:22:17	2004-10-15	14:22:17

CURRENT_TIMESTAMP dan SYSDATE adalah sinonim dari NOW(). Sedangkan CURRENT_DATE dan CURRENT_TIME adalah sinonim untuk CURDATE() dan CURTIME().

Catatan :
NOW() bukan tipe data

Fungsi seperti NOW() dan CURDATE() biasanya digunakan dalam statement CREATE TABEL sebagai nilai default.

```
mysql> CREATE TABEL testtbl (dt DATETIME DEFAULT NOW( ));
```

Apabila perintah ini anda tuliskan, maka akan terjadi **error**. Maksud sebenarnya dari perintah ini adalah memberi nilai default dari kolom dt. Kesalahan terjadi karena nilai default pada MySQL harus konstanta. Jika anda akan menset suatu kolom ke tanggal dan waktu sekarang pada saat pembuatan tabel, gunakan TIMESTAMP atau DATETIME dan set nilai awal ketika anda memasukkan data.

1.5 Mengambil hanya bagian dari tanggal atau waktu

MySQL menyediakan beberapa pilihan untuk memisahkan tanggal atau waktu. Hal ini bisa dilakukan dengan menggunakan fungsi DATE_FORMAT() dan TIME_FORMAT().

```
mysql> SELECT dt, DATE_FORMAT(dt,'%Y') AS year,
-> DATE_FORMAT(dt,'%d') AS day, TIME_FORMAT(dt,'%H') AS hour,
-> TIME_FORMAT(dt,'%s') AS second FROM datetime_val;
```

dt	year	day	hour	second
1970-01-01 00:00:00	1970	01	00	00
1987-03-05 12:30:15	1987	05	12	15
1999-12-31 09:00:00	1999	31	09	00
2000-06-04 15:45:30	2000	04	15	30

Fungsi `DATE_FORMAT()` dan `TIME_FORMAT()` juga dapat digunakan untuk mengambil lebih dari satu nilai. Misalnya untuk mengambil seluruh tanggal atau waktu dari `DATETIME`, dapat digunakan perintah berikut ini :

```
mysql> SELECT dt, DATE_FORMAT(dt,'%Y-%m-%d') AS 'date part',
-> TIME_FORMAT(dt,'%T') AS 'time part' FROM datetime_val;
```

dt	date part	time part
1970-01-01 00:00:00	1970-01-01	00:00:00
1987-03-05 12:30:15	1987-03-05	12:30:15
1999-12-31 09:00:00	1999-12-31	09:00:00
2000-06-04 15:45:30	2000-06-04	15:45:30

Salah satu keuntungan menggunakan fungsi-fungsi diatas adalah bahwa anda dapat menampilkan nilai yang diambil dalam bentuk berbeda dari bentuk aslinya. Perintah dibawah ini menunjukkan bagaimana melakukannya :

```
mysql> SELECT ts, DATE_FORMAT(ts,'%M %e, %Y') AS 'descriptive date',
-> TIME_FORMAT(ts,'%H:%i') AS 'hours/minutes' FROM timestamp_val;
```

ts	descriptive date	hours/minutes
19700101000000	January 1, 1970	00:00
19870305123015	March 5, 1987	12:30
19991231090000	December 31, 1999	09:00
20000604154530	June 4, 2000	15:45

1.6 Mengambil Tanggal atau Waktu menggunakan Fungsi Pengambilan Komponen

MySQL mempunyai banyak fungsi untuk mengambil bagian dari tanggal dan waktu. Beberapa diantaranya ditunjukkan pada tabel dibawah ini. Fungsi-fungsi yang berhubungan dengan tanggal bisa mengolah data dengan tipe `DATE`, `DATETIME` atau `TIMESTAMP`. Sedang fungsi-fungsi yang berhubungan dengan waktu bisa mengolah data dengan tipe `TIME`, `DATETIME` atau `TIMESTAMP`.

Fungsi

Nilai kembalian

<code>YEAR()</code>	Year of date
<code>MONTH()</code>	Month number (1..12)
<code>MONTHNAME()</code>	Month name (January..December)
<code>DAYOFMONTH()</code>	Day of month (1..31)
<code>DAYNAME()</code>	Day of week (Sunday..Saturday)
<code>DAYOFWEEK()</code>	Day of week (1..7 for Sunday..Saturday)
<code>WEEKDAY()</code>	Day of week (0..6 for Monday..Sunday)
<code>DAYOFYEAR()</code>	Day of year (1..366)
<code>HOURL()</code>	Hour of time (0..23)
<code>MINUTE()</code>	Minute of time (0..59)
<code>SECOND()</code>	Second of time (0..59)

Berikut adalah contoh pemakaian fungsi-fungsi diatas :

```
mysql> SELECT dt, YEAR(dt), DAYOFMONTH(dt), HOUR(dt), SECOND(dt)
-> FROM datetime_val;
```

dt	YEAR(dt)	DAYOFMONTH(dt)	HOUR(dt)	SECOND(dt)
1970-01-01 00:00:00	1970	1	0	0
1987-03-05 12:30:15	1987	5	12	15
1999-12-31 09:00:00	1999	31	9	0
2000-06-04 15:45:30	2000	4	15	30

Fungsi seperti YEAR() atau DAYOFMONTH() akan mengambil substring yang ada dalam suatu data tanggal atau waktu. MySQL juga menyediakan fungsi-fungsi yang mengambil nilai yang bukan merupakan substring dari data tanggal atau waktu. Salah satu contohnya adalah DAYOFYEAR() :

```
mysql> SELECT d, DAYOFYEAR(d) FROM date_val;
```

d	DAYOFYEAR(d)
1864-02-28	59
1900-01-15	15
1987-03-05	64
1999-12-31	365
2000-06-04	156

Contoh lain adalah DAYNAME() yang mengambil nama hari. Berikut adalah contoh pemakaiannya :

```
mysql> SELECT d, DAYNAME(d), LEFT(DAYNAME(d),3) FROM date_val;
```

d	DAYNAME(d)	LEFT(DAYNAME(d),3)
1864-02-28	Sunday	Sun
1900-01-15	Monday	Mon
1987-03-05	Thursday	Thu
1999-12-31	Friday	Fri
2000-06-04	Sunday	Sun

Untuk mendapatkan nama hari yang ditampilkan dalam angka, dapat digunakan DAYOFWEEK() atau WEEKDAY(). Tapi yang perlu diperhatikan adalah jangkauan nilai kembalian dari masing-masing fungsi. DAYOFWEEK() mengembalikan nilai antara 1 – 7 yang berhubungan dengan Minggu hingga Sabtu. Sedang WEEKDAY() mengembalikan nilai 0 – 6 yang berhubungan dengan Senin hingga Minggu.

```
mysql> SELECT d, DAYNAME(d), DAYOFWEEK(d), WEEKDAY(d) FROM date_val;
```

d	DAYNAME(d)	DAYOFWEEK(d)	WEEKDAY(d)
1864-02-28	Sunday	1	6
1900-01-15	Monday	2	0
1987-03-05	Thursday	5	3
1999-12-31	Friday	6	4

2000-06-04	Sunday	1	6
------------	--------	---	---

Cara lain untuk mendapatkan bagian dari data waktu adalah dengan menggunakan fungsi `EXTRACT()`.

```
mysql> SELECT dt, EXTRACT(DAY FROM dt), EXTRACT(HOUR FROM dt)
-> FROM datetime_val;
```

dt	EXTRACT(DAY FROM dt)	EXTRACT(HOUR FROM dt)
1970-01-01 00:00:00	1	0
1987-03-05 12:30:15	5	12
1999-12-31 09:00:00	31	9
2000-06-04 15:45:30	4	15

Berikut ini adalah contoh fungsi yang digunakan untuk mengambil bagian dari data tanggal sekarang yaitu tahun, bulan, tanggal dan nama hari dari fungsi `CURDATE()` atau `NOW()`:

```
mysql> SELECT CURDATE( ), YEAR(CURDATE( )) AS year,
-> MONTH(CURDATE( )) AS month, MONTHNAME(CURDATE( )) AS monthname,
-> DAYOFMONTH(CURDATE( )) AS day, DAYNAME(CURDATE( )) AS dayname;
```

CURDATE()	year	month	monthname	day	dayname
2002-07-15	2002	7	July	15	Monday

Bagian dari data waktu sekarang yang berupa jam, menit dan detik dapat diambil dengan menggunakan fungsi-fungsi `HOUR()`, `MINUTE()` dan `SECOND()`. Data waktunya sendiri diambil dari fungsi `CURTIME()` atau `NOW()`.

```
mysql> SELECT NOW( ), HOUR(NOW( )) AS hour, MINUTE(NOW( )) AS minute,
SECOND(NOW( )) AS second;
```

NOW()	hour	minute	second
2002-07-15 11:21:12	11	21	12

1.7 Menguraikan Tanggal dan Waktu menggunakan fungsi-fungsi string

Jika anda memasukkan data tanggal atau waktu pada suatu fungsi string, MySQL akan memperlakukannya sebagai suatu string, yang berarti anda dapat mengambil substringsnya. Oleh karena itu cara lain untuk mengambil bagian data tanggal dan waktu adalah menggunakan fungsi-fungsi string seperti `LEFT()` atau `MID()`.

```
mysql> SELECT dt, LEFT(dt,4) AS year, MID(dt,9,2) AS day,
-> RIGHT(dt,2) AS second FROM datetime_val;
```

dt	year	day	second
1970-01-01 00:00:00	1970	01	00
1987-03-05 12:30:15	1987	05	15
1999-12-31 09:00:00	1999	31	00

2000-06-04 15:45:30	2000	04	30
---------------------	------	----	----

Anda dapat mengambil data tanggal atau waktu dari DATETIME dengan menggunakan fungsi string seperti LEFT() atau RIGHT() :

```
mysql> SELECT dt, LEFT(dt,10) AS date, RIGHT(dt,8) AS time
-> FROM datetime_val;
```

dt	date	time
1970-01-01 00:00:00	1970-01-01	00:00:00
1987-03-05 12:30:15	1987-03-05	12:30:15
1999-12-31 09:00:00	1999-12-31	09:00:00
2000-06-04 15:45:30	2000-06-04	15:45:30

Teknik yang sama juga bisa diterapkan pada tipe data TIMESTAMP. Tetapi karena tipe data ini tidak mempunyai pemisah karakter, maka pemakaian fungsi LEFT() dan RIGHT() agak berbeda, demikian juga dengan keluarannya.

```
mysql> SELECT ts, LEFT(ts,8) AS date, RIGHT(ts,6) AS time
-> FROM timestamp_val;
```

ts	date	time
19700101000000	19700101	000000
19870305123015	19870305	123015
19991231090000	19991231	090000
20000604154530	20000604	154530

1.8 Merubah Data Tanggal dan Waktu Menggunakan DATE_FORMAT() dan TIME_FORMAT()

Menggabungkan suatu data dengan bagian dari data tanggal bisa dilakukan dengan mengambil bagian data yang diinginkan dan mengganti yang lainnya dengan data baru. Misalnya untuk mencari awal tanggal dari suatu bulan, bisa digunakan fungsi DATE_FORMAT() untuk mengambil data tanggal dan bulan dari data tanggal kemudian menggabungkannya dengan 01 :

```
mysql> SELECT d, DATE_FORMAT(d,'%Y-%m-01') FROM date_val;
```

d	DATE_FORMAT(d,'%Y-%m-01')
1864-02-28	1864-02-01
1900-01-15	1900-01-01
1987-03-05	1987-03-01
1999-12-31	1999-12-01
2000-06-04	2000-06-01

TIME_FORMAT() dapat digunakan dengan cara yang sama :

```
mysql> SELECT t1, TIME_FORMAT(t1,'%H:%i:00') FROM time_val;
```

t1	TIME_FORMAT(t1,'%H:%i:00')
15:00:00	15:00:00
05:01:30	05:01:00
12:30:20	12:30:00

1.9 Merubah Data Tanggal dan Waktu menggunakan Fungsi CONCAT()

Cara lain untuk merubah data tanggal dan waktu adalah menggunakan fungsi CONCAT(). Tapi metoda ini tidak lebih baik dari pada dengan menggunakan DATE_FORMAT() dan sering memberikan hasil yang berbeda :

```
mysql> SELECT d, CONCAT(YEAR(d),'-',MONTH(d),'-01') FROM date_val;
```

d	CONCAT(YEAR(d),'-',MONTH(d),'-01')
1864-02-28	1864-2-01
1900-01-15	1900-1-01
1987-03-05	1987-3-01
1999-12-31	1999-12-01
2000-06-04	2000-6-01

Perlu dicatat bahwa nilai bulan pada beberapa data tanggal hanya mempunyai satu angka. Untuk memastikan bahwa bulan memiliki dua angka, dapat digunakan LPAD() untuk menambah angka nol didepan angka bulan yang hanya mempunyai satu angka.

```
mysql> SELECT d, CONCAT(YEAR(d),'-',LPAD(MONTH(d),2,'0'),'01')
-> FROM date_val;
```

d	CONCAT(YEAR(d),'-',LPAD(MONTH(d),2,'0'),'01')
1864-02-28	1864-02-01
1900-01-15	1900-01-01
1987-03-05	1987-03-01
1999-12-31	1999-12-01
2000-06-04	2000-06-01

Untuk merubah data waktu sehingga bagian detik nilainya 00, maka yang harus dilakukan adalah mengambil bagian jam dan menit, kemudian menggabungkannya dengan nilai 00 tersebut. Hal ini bisa dilakukan dengan dua cara yaitu menggunakan TIME_FORMAT atau CONCAT().

```
mysql> SELECT t1, TIME_FORMAT(t1,'%H:%i:00') AS method1,
-> CONCAT(LPAD(HOUR(t1),2,'0'),':',LPAD(MINUTE(t1),2,'0'),'00') AS
method2 FROM time_val;
```

t1	method1	method2
15:00:00	15:00:00	15:00:00
05:01:30	05:01:00	05:01:00
12:30:20	12:30:00	12:30:00

1.10 Menggabungkan data Tanggal dan Waktu menjadi Data Tanggal-Waktu

Menggabungkan data tanggal dan waktu menjadi satu data tanggal-waktu adalah masalah penggabungan keduanya yang dipisahkan dengan spasi.

```
mysql> SET @d = '2002-02-28';
mysql> SET @t = '13:10:05';
mysql> SELECT @d, @t, CONCAT(@d, ' ', @t);
```

@d	@t	CONCAT(@d, ' ', @t)
2002-02-28	13:10:05	2002-02-28 13:10:05

1.11 Merubah data waktu menjadi dalam bentuk detik dan sebaliknya

Fungsi `TIME_TO_SEC()` merubah data waktu menjadi data dalam detik. Sedang fungsi `SEC_TO_TIME()` melakukan hal yang sebaliknya. Query berikut ini menunjukkan cara menggunakan kedua fungsi tersebut.

```
mysql> SELECT t1, TIME_TO_SEC(t1) AS 'TIME to seconds',
-> SEC_TO_TIME(TIME_TO_SEC(t1)) AS 'TIME to seconds to TIME'
-> FROM time_val;
```

t1	TIME to seconds	TIME to seconds to TIME
15:00:00	54000	15:00:00
05:01:30	18090	05:01:30
12:30:20	45020	12:30:20

Untuk merubah data waktu menjadi menit atau hari, dapat dilakukan dengan cara membagi data dalam detik dengan nilai yang semestinya. Misalnya untuk merubah menjadi data dalam menit, maka data dalam detik harus dibagi dengan 60.

```
mysql> SELECT t1, TIME_TO_SEC(t1) AS 'seconds',
-> TIME_TO_SEC(t1)/60 AS 'minutes', TIME_TO_SEC(t1)/(60*60) AS 'hours',
-> TIME_TO_SEC(t1)/(24*60*60) AS 'days' FROM time_val;
```

t1	seconds	minutes	hours	days
15:00:00	54000	900.00	15.00	0.62
05:01:30	18090	301.50	5.03	0.21
12:30:20	45020	750.33	12.51	0.52

Untuk menampilkan data dalam bentuk integer dan bukan floating point, maka dapat digunakan fungsi `FLOOR()`.

```
mysql> SELECT t1, TIME_TO_SEC(t1) AS 'seconds',
-> FLOOR(TIME_TO_SEC(t1)/60) AS 'minutes',
-> FLOOR(TIME_TO_SEC(t1)/(60*60)) AS 'hours',
-> FLOOR(TIME_TO_SEC(t1)/(24*60*60)) AS 'days'
-> FROM time_val;
```

t1	seconds	minutes	hours	days
15:00:00	54000	900	15	0
05:01:30	18090	301	5	0
12:30:20	45020	750	12	0

15:00:00	54000	900	15	0
05:01:30	18090	301	5	0
12:30:20	45020	750	12	0

Jika data tanggal dan waktu (DATETIME) dimasukkan pada fungsi `TIME_TO_SEC()`, maka yang akan diambil hanya bagian waktu saja sedang bagian tanggal akan dibuang. Sehingga ini bisa digunakan untuk mengambil data waktu dari data bertipe DATETIME dan TIMESTAMP.

```
mysql> SELECT dt, TIME_TO_SEC(dt) AS 'time part in seconds',
-> SEC_TO_TIME(TIME_TO_SEC(dt)) AS 'time part as TIME' FROM datetime_val;
```

dt	time part in seconds	time part as TIME
1970-01-01 00:00:00	0	00:00:00
1987-03-05 12:30:15	45015	12:30:15
1999-12-31 09:00:00	32400	09:00:00
2000-06-04 15:45:30	56730	15:45:30

```
mysql> SELECT ts, TIME_TO_SEC(ts) AS 'time part in seconds',
-> SEC_TO_TIME(TIME_TO_SEC(ts)) AS 'time part as TIME' FROM timestamp_val;
```

ts	time part in seconds	time part as TIME
19700101000000	0	00:00:00
19870305123015	45015	12:30:15
19991231090000	32400	09:00:00
20000604154530	56730	15:45:30

1.12 Merubah Data dari Tanggal ke Hari dan Sebaliknya

Data tanggal dapat dirubah menjadi hari dengan `TO_DAYS()` dan `FROM_DAYS()`. Data tanggal dan waktu juga bisa dirubah ke hari dengan menghilangkan bagian waktu.

`TO_DAYS()` merubah tanggal ke jumlah hari, sedang `FROM_DAYS()` melakukan hal yang sebaliknya.

```
mysql> SELECT d, TO_DAYS(d) AS 'DATE to days',
-> FROM_DAYS(TO_DAYS(d)) AS 'DATE to days to DATE' FROM date_val;
```

d	DATE to days	DATE to days to DATE
1864-02-28	680870	1864-02-28
1900-01-15	693975	1900-01-15
1987-03-05	725800	1987-03-05
1999-12-31	730484	1999-12-31
2000-06-04	730640	2000-06-04

Ketika menggunakan `TO_DAYS()`, sebaiknya tidak menggunakan tanggal sebelum permulaan kalender Gregorian (1582). Jika anda memasukkan data tanggal dan waktu kedalam fungsi `TO_DAYS()`, maka hanya bagian tanggal saja yang diambil, sedang bagian waktu akan dibuang. Ini menyediakan cara lain untuk mengambil bagian tanggal dari data bertipe DATETIME dan TIMESTAMP.

```
mysql> SELECT dt, TO_DAYS(dt) AS 'date part in days',
-> FROM_DAYS(TO_DAYS(dt)) AS 'date part as DATE' FROM datetime_val;
```

dt	date part in days	date part as DATE
1970-01-01 00:00:00	719528	1970-01-01
1987-03-05 12:30:15	725800	1987-03-05
1999-12-31 09:00:00	730484	1999-12-31
2000-06-04 15:45:30	730640	2000-06-04

```
mysql> SELECT ts, TO_DAYS(ts) AS 'date part in days',
-> FROM_DAYS(TO_DAYS(ts)) AS 'date part as DATE' FROM timestamp_val;
```

ts	date part in days	date part as DATE
19700101000000	719528	1970-01-01
19870305123015	725800	1987-03-05
19991231090000	730484	1999-12-31
20000604154530	730640	2000-06-04

1.13. Konversi Antara Nilai Tanggal dan Waktu dan Detik

Penyelesaian

Fungsi UNIX_TIMESTAMP() dan FROM_TIMESTAMP() mengkonversi nilai DATETIME atau TIMESTAMP pada jangkauan dari 1970 hingga sekitar 2037 ke dan dari jumlah detik sejak permulaan tahun 1970. Konversi ke detik menawarkan presisi lebih tinggi untuk nilai tanggal dan waktu dari pada konversi ke tanggal.

Pebahasan

Ketika bekerja dengan nilai tanggal dan waktu, dapat digunakan TO_DAYS() dan FROM_DAYS() untuk mengkonversi nilai tanggal ke hari dan kembali ke tanggal. Untuk nilai tanggal yang terjadi tidak lebih awal dari 1970-01-01 00:00:00 GMT dan tidak lebih lambat dari sekitar 2037, dimungkinkan untuk mendapatkan presisi lebih tinggi dengan mengkonversi ke dan dari detik. Fungsi UNIX_DATETIME() mengkonversi nilai tanggal dan waktu dalam jangkauan ini ke jumlah detik yang telah berlalu sejak awal 1970, dan fungsi FROM_UNIXTIME() mengerjakan kebalikannya.

```
mysql> SELECT dt, UNIX_TIMESTAMP(dt) AS seconds,
-> FROM_UNIXTIME(UNIX_TIMESTAMP(dt)) AS timestamp FROM datetime_val;
```

dt	seconds	timestamp
1970-01-01 00:00:00	21600	1970-01-01 00:00:00
1987-03-05 12:30:15	541967415	1987-03-05 12:30:15
1999-12-31 09:00:00	946652400	1999-12-31 09:00:00
2000-06-04 15:45:30	960151530	2000-06-04 15:45:30

Hubungan antara UNIX dalam nama fungsi dan fakta bahwa jangkauan nilai yang dapat diaplikasikan dimulai dari 1970 adalah bahwa 1970-01-01 00:00:00 GMT menandai “**Masa Unix**”. Masa adalah waktu nol, atau titik referensi untuk menghitung waktu dalam sistem Unix. Tapi mengapa pada contoh diatas nilai UNIX_TIMESTAMP() adalah 21600. Mengapa tidak nol ? Apa yang terjadi ? Hal ini terjadi karena fakta bahwa MySQL server mengkonversi nilai ke

zona waktu nya sendiri ketika menampilkan nilai tersebut. Pada contoh ini server berada di zona waktu Amerika Bagian Tengah, yang enam jam (*yaitu 21600 detik*) disebelah barat GMT.

UNIX_TIMESTAMP() juga dapat mengkonversi nilai tanggal ke detik. Fungsi ini akan menganggap tanggal mempunyai nilai waktu 00:00:00.

```
mysql> SELECT CURDATE( ), FROM_UNIXTIME(UNIX_TIMESTAMP(CURDATE( )));
+-----+-----+
| CURDATE( ) | FROM_UNIXTIME(UNIX_TIMESTAMP(CURDATE( ))) |
+-----+-----+
| 2002-07-15 | 2002-07-15 00:00:00 |
+-----+-----+
```

1. 14. Menambah Interval Waktu ke Waktu

Penyelesaian

Menggunakan TIME_TO_SEC() akan merubah semua waktu untuk dinyatakan dalam detik, kemudian menambahkannya. Hasilnya akan disajikan dalam detik. Gunakan SEC_TO_TIME() jika akan mengembalikan ke nilai waktu.

Pembahasan

Alat utama untuk melaksanakan aritmetika waktu adalah TIME_TO_SEC() dan SEC_TO_TIME(), yang mengkonversi antara nilai waktu dan detik. Untuk menambah nilai interval dalam detik ke suatu nilai waktu, konversikan waktu ke detik sehingga kedua nilai disajikan dalam satuan yang sama, tambahkan kedua nilai dan konversikan hasil kembali ke waktu. Sebagai contoh, dua jam adalah 7200 detik ($2 * 60 * 60$), query berikut ini menambahkan dua jam ke setiap nilai t1 pada table time_val :

```
mysql> SELECT t1, SEC_TO_TIME(TIME_TO_SEC(t1) + 7200) AS 't1 plus 2 hours'
-> FROM time_val;
+-----+-----+
| t1          | t1 plus 2 hours |
+-----+-----+
| 15:00:00    | 17:00:00        |
| 05:01:30    | 07:01:30        |
| 12:30:20    | 14:30:20        |
+-----+-----+
```

Jika intervalnya sendiri dinyatakan sebagai waktu, dia juga harus dikonversi ke detik sebelum menambahkan kedua nilai. Contoh berikut ini menghitung jumlah dua nilai waktu pada tabel time_val :

```
mysql> SELECT t1, t2,
-> SEC_TO_TIME(TIME_TO_SEC(t1) + TIME_TO_SEC(t2)) AS 't1 + t2'
-> FROM time_val;
+-----+-----+-----+
| t1          | t2          | t1 + t2 |
+-----+-----+-----+
| 15:00:00    | 15:00:00    | 30:00:00 |
| 05:01:30    | 02:30:20    | 07:31:50 |
| 12:30:20    | 17:30:45    | 30:01:05 |
+-----+-----+-----+
```

Penting untuk mengetahui bahwa nilai waktu MySQL benar-benar menyajikan waktu yang sudah berlalu, bukan waktu hari (*time of day*), sehingga mereka tidak kembali ke 0 setelah mencapai 24

jam. Hal ini dapat dilihat pada baris pertama dan ketiga dari query diatas. Untuk menghasilkan nilai waktu hari, maka pastikan bahwa waktu akan kembali ke 0 setelah 24 jam dengan menggunakan operasi modulo sebelum mengkonversi nilai detik kembali ke nilai waktu. Jumlah detik dalam satu hari adalah $24 * 60 * 60$ atau 86400, sehingga untuk mengkonversi semua nilai detik tapi tetap berada pada jangkauan 24 jam, gunakan `MOD()` atau operator modulo `%` seperti contoh dibawah ini :

```
MOD(s,86400)
s % 86400
```

Kedua ekspresi diatas adalah sama saja. Menggunakan ekspresi pertama untuk menghitung contoh diatas akan menghasilkan :

```
mysql> SELECT t1, t2,
-> SEC_TO_TIME(MOD(TIME_TO_SEC(t1) + TIME_TO_SEC(t2), 86400)) AS 't1 + t2'
-> FROM time_val;
```

t1	t2	t1 + t2
15:00:00	15:00:00	06:00:00
05:01:30	02:30:20	07:31:50
12:30:20	17:30:45	06:01:05

Jangkauan nilai waktu yang diperbolehkan adalah -838:59:59 hingga 838:59:59 (yaitu -3020399 to 3020399). Ketika anda menambahkan dua buah waktu, bisa jadi anda akan mendapatkan hasil yang berada diluar jangkauan ini. Jika anda mencoba menyimpan nilai seperti itu, MySQL akan merubahnya ke nilai batas jangkauan terdekat.

1. 15. Menghitung Interval Antara Waktu

Penyelesaian

Konversikan kedua waktu ke detik dengan `TIME_TO_SEC()` dan menghitung selisihnya. Untuk menyajikan selisih sebagai waktu, konversikan hasilnya kembali dengan `SEC_TO_TIME()`.

Pembahasan

Menghitung interval antara dua waktu sama dengan menambahkan waktu, kecuali bahwa yang dihitung adalah selisih dan bukan jumlah. Sebagai contoh, untuk menghitung interval dalam detik antara pasangan nilai t1 dan t2, konversikan nilai-nilai dalam table time_val ke detik menggunakan `TIME_TO_SEC()`, kemudian hitung selisihnya. Untuk menampilkan selisih sebagai waktu, gunakan `SEC_TO_TIME()`. Berikut adalah query yang menunjukkan interval baik dalam detik maupun dalam waktu.

```
mysql> SELECT t1, t2,
-> TIME_TO_SEC(t2) - TIME_TO_SEC(t1) AS 'interval in seconds',
-> SEC_TO_TIME(TIME_TO_SEC(t2) - TIME_TO_SEC(t1)) AS 'interval as TIME'
-> FROM time_val;
```

t1	t2	interval in seconds	interval as TIME
15:00:00	15:00:00	0	00:00:00
05:01:30	02:30:20	-9070	-02:31:10
12:30:20	17:30:45	18025	05:00:25

Catatan bahwa interval bisa negaif, ketika t1 terjadi lebih lambat dari t2.

1.16. Memisahkan Interval Waktu menjadi Komponen-komponen

Penyelesaian

Pisahkan interval dengan menggunakan fungsi `HOUR()`, `MINUTE()`, dan `SECOND()`. Jika perhitungannya kompleks dan anda menggunakan interval dalam sebuah program, akan lebih mudah untuk menggunakan bahasa pemrograman untuk melaksanakan perhitungan yang sama.

Pembahasan

Untuk menyatakan interval waktu dalam nilai jam, menit, dan detik, dapat dilakukan dengan menghitung interval waktu dengan menggunakan fungsi `HOUR()`, `MINUTE()`, dan `SECOND()`. Misalnya, untuk menentukan komponen dari interval antara kolom `t1` dan `t2` pada table `time_val`, berikut adalah contoh querynya :

```
mysql> SELECT t1, t2,
-> SEC_TO_TIME(TIME_TO_SEC(t2) - TIME_TO_SEC(t1)) AS 'interval as TIME',
-> IF(SEC_TO_TIME(TIME_TO_SEC(t2) >= TIME_TO_SEC(t1)), '+', '-') AS sign,
-> HOUR(SEC_TO_TIME(TIME_TO_SEC(t2) - TIME_TO_SEC(t1))) AS hour,
-> MINUTE(SEC_TO_TIME(TIME_TO_SEC(t2) - TIME_TO_SEC(t1))) AS minute,
-> SECOND(SEC_TO_TIME(TIME_TO_SEC(t2) - TIME_TO_SEC(t1))) AS second
-> FROM time_val;
```

t1	t2	interval as TIME	sign	hour	minute	second
15:00:00	15:00:00	00:00:00	+	0	0	0
05:01:30	02:30:20	-02:31:10	-	2	31	10
12:30:20	17:30:45	05:00:25	+	5	0	25

Tetapi ini cukup membingungkan. Akan lebih baik menggunakan SQL untuk menghitung hanya interval dalam detik, kemudian gunakan bahasa API untuk memisahkan masing-masing interval kedalam komponennya. Rumus yang dipakai harus mempertimbangkan hasil negatif dan menghasilkan nilai integer untuk masing-masing komponen. Berikut ini adalah contoh `time_components()` yang ditulis dalam Python yang mencari nilai interval dalam detik dan mengembalikan empat nilai yang terdiri dari tanda nilai, diikuti dengan jam, menit, dan detik.

```
def time_components (time_in_secs):
    if time_in_secs < 0:
        sign = "-"
        time_in_secs = -time_in_secs
    else:
        sign = ""
    hours = int (time_in_secs / 3600)
    minutes = int ((time_in_secs / 60)) % 60
    seconds = time_in_secs % 60
    return (sign, hours, minutes, seconds)
```

Kamu mungkin menggunakan komponen waktu() seperti program ini:

```
query = "SELECT t1, t2, TIME_TO_SEC(t2) - TIME_TO_SEC(t1) FROM time_val"
cursor = conn.cursor ( )
cursor.execute (query)
for (t1, t2, interval) in cursor.fetchall ( ):
    (sign, hours, minutes, seconds) = time_components (interval)
    print "t1 = %s, t2 = %s, interval = %s%d h, %d m, %d s" \
    % (t1, t2, sign, hours, minutes, seconds)
cursor.close ( )
```


Program diatas menghasilkan keluaran sebagai berikut :

```
t1 = 15:00:00, t2 = 15:00:00, interval = 0 h, 0 m, 0 s
t1 = 05:01:30, t2 = 02:30:20, interval = -2 h, 31 m, 10 s
t1 = 12:30:20, t2 = 17:30:45, interval = 5 h, 0 m, 25 s
```

Contoh diatas menggambarkan prinsip umum bahwa akan lebih berguna jika mengeluarkan query dari sebuah program. Akan lebih mudah melaksanakan perhitungan yang kompleks untuk dinyatakan dalam SQL hanya dengan menggunakan query sederhana dan memproses hasil lebih lanjut menggunakan bahasa API.

1.17. Mencari Hari untuk suatu Tanggal

Penyelesaian

Gunakan fungsi DAYNAME () .

Pembahasan

Untuk menentukan nama hari untuk suatu tanggal, gunakan DAYNAME () :

```
mysql> SELECT CURDATE( ), DAYNAME(CURDATE( ));
+-----+-----+
| CURDATE( ) | DAYNAME(CURDATE( )) |
+-----+-----+
| 2002-07-15 | Monday               |
+-----+-----+
```

DAYNAME () sering berguna dalam hubungannya dengan teknik lain yang berhubungan dengan tanggal . Misalnya, untuk mencari hari pada awal bulan, gunakan ekspresi first-of-month dari penjelasan sebelum ini dan gunakan sebagai argumen pada fungsi DAYNAME () :

```
mysql> SET @d = CURDATE( );
mysql> SET @first = DATE_SUB(@d,INTERVAL DAYOFMONTH(@d)-1 DAY);
mysql> SELECT @d AS 'starting date',
-> @first AS '1st of month date',
-> DAYNAME(@first) AS '1st of month day';
+-----+-----+-----+
| starting date | 1st of month date | 1st of month day |
+-----+-----+-----+
| 2002-07-15   | 2002-07-01       | Monday           |
+-----+-----+-----+
```

1.18. Mencari Tanggal untuk Hari pada Minggu ini

Penyelesaian

Cari jumlah hari antara hari mulai dan hari yang diinginkan, kemudian geser tanggal sejauh jumlah hari tersebut.

Pembahasan

Bagian ini dan selanjutnya menjelaskan bagaimana untuk mengkonversi satu tanggal ke tanggal lain ketika tanggal tujuan ditentukan dalam hari. Untuk menyelesaikan permasalahan semacam itu harus diketahui nilai hari. Misalnya, jika anda ingin mengetahui tanggal berapa hari Selasa minggu ini. Jika hari ini adalah Senin, maka anda harus menambah satu hari ke CURDATE () , tapi jika hari ini adalah Rabu, anda harus mengurangi dengan satu hari.

MySQL menyediakan dua fungsi yang berguna dalam hal ini. `DAYOFWEEK()` memperlakukan Minggu sebagai hari pertama dalam satu minggu dan mengembalikan 1 hingga 7 untuk Minggu hingga Sabtu. `WEEKDAY()` memperlakukan Senin sebagai hari pertama dalam satu minggu dan mengembalikan 0 hingga 6 untuk Senin hingga Minggu. Operasi lain yang berhubungan dengan hari dalam satu minggu melibatkan penentuan nama hari. `DAYNAME()` dapat digunakan untuk hal ini. Perhitungan yang menentukan satu hari dalam satu minggu dari hari lainnya tergantung pada hari dimana anda memulai demikian juga hari yang akan anda tuju. Akan lebih mudah untuk menggeser tanggalnya dulu ke titik relatif yang diketahui ke awal minggu, kemudian geser kedepan :

- Geser tanggal kembali dengan nilai `DAYOFWEEK()` nya, yang selalu menghasilkan tanggal untuk hari Sabtu minggu lalu.
- Tambah satu hari ke hasil ini untuk mencapai tanggal untuk hari Minggu, dan sebagainya.

Dalam SQL, operasi ini dapat dinyatakan sebagai berikut untuk tanggal `d`, dimana `n` adalah 1 hingga 7 untuk menghasilkan tanggal untuk Minggu hingga Sabtu :

```
DATE_ADD( DATE_SUB( d, INTERVAL DAYOFWEEK(d) DAY ), INTERVAL n DAY )
```

Eksprei diatas membagi frasi “geser kebelakan ke Sabtu” dan “geser kedepan” menjadi operasi terpisah, tapi karena interval untuk baik `DATE_SUB()` dan `DATE_ADD()` adalah dalam tanggal, ekspresi ini dapat dipersingkat menjadi hanya `DATE_ADD()` :

```
DATE_ADD( d, INTERVAL n-DAYOFWEEK(d) DAY )
```

Jika kita menggunakan rumus ini ke table `date_val`, menggunakan `n` sebesar 1 untuk Minggu dan 7 untuk Sabtu untuk mencari awal dan akhir minggu, kita akan mendapatkan hasil sebagai berikut :

```
mysql> SELECT d, DAYNAME(d) AS day,
-> DATE_ADD(d,INTERVAL 1-DAYOFWEEK(d) DAY) AS Sunday,
-> DATE_ADD(d,INTERVAL 7-DAYOFWEEK(d) DAY) AS Saturday
-> FROM date_val;
```

d	day	Sunday	Saturday
1864-02-28	Sunday	1864-02-28	1864-03-05
1900-01-15	Monday	1900-01-14	1900-01-20
1987-03-05	Thursday	1987-03-01	1987-03-07
1999-12-31	Friday	1999-12-26	2000-01-01
2000-06-04	Sunday	2000-06-04	2000-06-10

1.19 Mencari Tanggal untuk Hari bukan pada Minggu ini

Penyelesaian

Cari tanggal untuk hari itu pada minggu itu, kemudian geser hasilnya ke minggu yang dituju.

Pembahasan

Mencari tanggal suatu hari bukan pada minggu ini adalah masalah yang terdiri dari geser hari dalam minggu ini dan geser minggu. Operasi ini dapat dilakukan dalam urutan manapun sebab jumlah pergeseran dalam minggu adalah sama apakah anda menggeser tanggalnya ke minggu yang dituju lebih dahulu atau belakangan. Sebagai contoh untuk menghitung Rabu dalam suatu minggu

dengan menggunakan rumus didepan, n adalah 4. Untuk menghitung tanggal untuk Rabu dua minggu yang lalu, anda dapat melakukan pergeseran hari lebih dulu, seperti berikut ini :

```
mysql> SET @target =
-> DATE_SUB(DATE_ADD(CURDATE( ),INTERVAL 4-DAYOFWEEK(CURDATE( )) DAY),
-> INTERVAL 14 DAY);
mysql> SELECT CURDATE( ), @target, DAYNAME(@target);
+-----+-----+-----+
| CURDATE( ) | @target | DAYNAME(@target) |
+-----+-----+-----+
| 2002-07-15 | 2002-07-03 | Wednesday |
+-----+-----+-----+
```

Atau anda dapat melakukan pergeseran minggu lebih dulu :

```
mysql> SET @target =
-> DATE_ADD(DATE_SUB(CURDATE( ),INTERVAL 14 DAY),
-> INTERVAL 4-DAYOFWEEK(CURDATE( )) DAY);
mysql> SELECT CURDATE( ), @target, DAYNAME(@target);
+-----+-----+-----+
| CURDATE( ) | @target | DAYNAME(@target) |
+-----+-----+-----+
| 2002-07-15 | 2002-07-03 | Wednesday |
+-----+-----+-----+
```

Beberapa aplikasi membutuhkan penentuan tanggal seperti n-th. Sebagai contoh, jika anda mengurus penggajian dimana hari pembayaran adalah hari Kamis ke 2 (2nd) dan ke 4 (4th) setiap bulan, maka anda harus tahu tanggal berapakah itu. Salah satu cara untuk mengetahui hal ini untuk bulan tertentu adalah dengan mencari tanggal awal bulan dan menggesernya kedepan. Cukup mudah untuk menggeser tanggal ke Kamis minggu itu. Yang perlu diketahui adalah untuk mencari berapa minggu kita harus menggeser kedepan untuk mencapai Kamis kedua (2nd) dan keempat (4th). Jika awal bulan terjadi pada hari dari Minggu hingga Kamis, anda harus menggeser kedepan satu minggu untuk mencapai Kamis kedua. Jika awal bulan terjadi pada Jum'at atau setelah itu, anda harus menggeser kedepan dua minggu. Kamis keempat terletak pada dua minggu setelah itu.

Kode Perl berikut ini mengimplementasikan logika untuk mencari semua hari pembayaran pada tahun 2002. Kode ini melaksanakan perulangan yang menghasilkan tanggal awal bulan untuk semua bulan pada tahun itu. Untuk setiap bulang, kode ini akan mengeluarkan query yang menentukan tanggal dari Kamis kedua dan keempat :

```
my $year = 2002;
print "MM/CCYY 2nd Thursday 4th Thursday\n";
foreach my $month (1..12)
{
my $first = sprintf ("%04d-%02d-01", $year, $month);
my ($thu2, $thu4) = $dbh->selectrow_array (qq{
SELECT
DATE_ADD(
DATE_ADD(?,INTERVAL 5-DAYOFWEEK(?) DAY),
INTERVAL IF(DAYOFWEEK(?) <= 5, 7, 14) DAY),
DATE_ADD(
DATE_ADD(?,INTERVAL 5-DAYOFWEEK(?) DAY),
INTERVAL IF(DAYOFWEEK(?) <= 5, 21, 28) DAY)
}, undef, $first, $first, $first, $first, $first, $first);
printf "%02d/%04d %s %s\n", $month, $year, $thu2, $thu4;
```

```
}
```

Keluarannya akan terlihat seperti berikut ini :

```
MM/CCYY 2nd Thursday 4th Thursday
01/2002 2002-01-10 2002-01-24
02/2002 2002-02-14 2002-02-28
03/2002 2002-03-14 2002-03-28
04/2002 2002-04-11 2002-04-25
05/2002 2002-05-09 2002-05-23
06/2002 2002-06-13 2002-06-27
07/2002 2002-07-11 2002-07-25
08/2002 2002-08-08 2002-08-22
09/2002 2002-09-12 2002-09-26
10/2002 2002-10-10 2002-10-24
11/2002 2002-11-14 2002-11-28
12/2002 2002-12-12 2002-12-26
```

1.20 Melakukan Perhitungan Tahun Kabisat

Penyelesaian

Anda harus tahu bagaimana untuk mengetahui apakah suatu tahun adalah tahun kabisat atau bukan dan memasukkan hasilnya kedalam perhitungan.

Pembahasan

Perhitungan tanggal dipersulit dengan fakta bahwa tanggal tidak mempunyai jumlah hari yang sama dan Februari mempunyai satu hari ekstra pada tahun kabisat. Bagian ini menunjukkan bagaimana untuk menentukan apakah suatu tanggal terletak pada tahun kabisat atau tidak dan bagaimana untuk memasukkan tahun kabisat dalam perhitungan ketika menentukan panjang tahun atau bulan.

Menentukan Apakah suatu Tanggal Terletak pada Tahun Kabisat

Untuk menentukan apakah suatu tanggal d terletak pada tahun kabisat, carilah komponen tahun dengan menggunakan YEAR() dan ceklah hasilnya. Aturan umum untuk mengecek tahun kabisat adalah bahwa $\text{YEAR}(d) \% 4 = 0$. Tapi hal ini tidak betul.

Tahun yang dapat dianggap sebagai tahun kabisat adalah tahun yang memenuhi ketentuan berikut ini :

- Tahun harus dapat dibagi dengan empat.
- Tahun tidak dapat dibagi 100, kecuali jika juga dapat dibagi 400.

Arti dari aturan kedua adalah bahwa tahun-tahun dalam satu abad bukan tahun kabisat, kecuali setiap abad. Dalam SQL, anda dapat menyajikan hal ini sebagai berikut :

```
(YEAR(d) % 4 = 0) AND ((YEAR(d) % 100 != 0) OR (YEAR(d) % 400 = 0))
```

Apabila rumus diatas dilaksanakan pada table date_val, maka akan menghasilkan hasil berikut ini :

```
mysql> SELECT d, YEAR(d) % 4 = 0 AS "rule-of-thumb test",
-> (YEAR(d) % 4 = 0) AND ((YEAR(d) % 100 != 0) OR (YEAR(d) % 400 = 0))
-> AS "complete test" FROM date_val;
+-----+-----+-----+
| d          | rule-of-thumb test | complete test |
+-----+-----+-----+
```

1864-02-28	1	1
1900-01-15	1	0
1987-03-05	0	0
1999-12-31	0	0
2000-06-04	1	1

Karena pengecekan tahun kabisat harus mengecek abad, maka yang dibutuhkan adalah nilai tahun dengan empat digit. Tahun dengan dua digit akan mendapatkan hasil yang mendua. Jika anda bekerja dengan nilai tanggal dalam sebuah program, anda dapat melakukan pengecekan tahun kabisat dengan bahasa API dari pada dengan SQL. Ambil bagian tahun dari sebuah tanggal, kemudian ceklah tahun itu. Jika bahasa API dapat melakukan konversi dari string ke angka secara otomatis, pengecekan akan lebih mudah. Kalau tidak, anda harus mengkonversi nilai tahun ke angka sebelum pengecekan :

Pada Perl dan PHP, pengecekan tahun kabisat adalah sebagai berikut :

```
$year = substr ($date, 0, 4);
$is_leap = ($year % 4 == 0) && ($year % 100 != 0 || $year % 400 == 0);
```

Sintaks untuk Python juga mirip, meskipun operasi konversi tipe sangat diperlukan :

```
year = int (date[0:4])
is_leap = (year % 4 == 0) and (year % 100 != 0 or year % 400 == 0)
```

Konversi tipe pada Java juga diperlukan :

```
int year = Integer.valueOf (date.substring (0, 4)).intValue ( );
boolean is_leap = (year % 4 == 0) && (year % 100 != 0 || year % 400 == 0);
```

Menggunakan Tahun Kabisat untuk Menghitung Panjang Tahun

Tahun biasanya berjumlah 365 hari, tapi tahun kabisat mempunyai satu hari ekstra. Untuk menentukan panjang suatu tahun dimana suatu tanggal terdapat, anda dapat menggunakan salah satu pengecekan tahun kabisat untuk mengetahui :

```
$year = substr ($date, 0, 4);
$is_leap = ($year % 4 == 0) && ($year % 100 != 0 || $year % 400 == 0);
$days_in_year = ($is_leap ? 366 : 365);
```

Cara lain untuk menghitung panjang suatu tahun adalah dengan menghitung tanggal dari hari terakhir pada tahun itu memasukkannya dalam fungsi DAYOFYEAR () :

```
mysql> SET @d = '2003-04-13';
mysql> SELECT DAYOFYEAR (DATE_FORMAT (@d, '%Y-12-31'));
+-----+
| DAYOFYEAR (DATE_FORMAT (@d, '%Y-12-31')) |
+-----+
| 365 |
+-----+

mysql> SET @d = '2004-04-13';
mysql> SELECT DAYOFYEAR (DATE_FORMAT (@d, '%Y-12-31'));
+-----+
| DAYOFYEAR (DATE_FORMAT (@d, '%Y-12-31')) |
+-----+
| 366 |
+-----+
```

+-----+

Menggunakan Tahun Kabisat untuk Menghitung Panjang Bulan

Kita sudah mengemukakan bagaimana untuk menentukan jumlah hari dalam satu bulan menggunakan pergeseran tanggal untuk mencari hari terakhir pada bulan itu. Pengecekan tahun kabisat menyediakan cara alternatif untuk melakukan hal yang sama. Semua bulan kecuali Februari mempunyai jumlah hari yang tetap, sehingga dengan menguji bagian bulan dari suatu tanggal, anda dapat mengatakan berapa panjang bulan itu. Anda juga dapat mengatakan berapa jumlah hari pada suatu bulan Februari jika anda mengetahui apakah bulan itu terjadi pada tahun kabisat atau tidak. Jumlah hari dalam satu bulan dapat dinyatakan dalam SQL sebagai berikut :

```
mysql> SELECT d, ELT(MONTH(d), 31,
-> IF((YEAR(d)%4 = 0) AND ((YEAR(d)%100 != 0) OR (YEAR(d)%400 = 0)),29,28),
-> 31,30,31,30,31,31,30,31,30,31) AS 'days in month' FROM date_val;
```

d	days in month
1864-02-28	29
1900-01-15	31
1987-03-05	31
1999-12-31	31
2000-06-04	30

Fungsi ELT() mengevaluasi argumen pertamanya untuk menentukan nilai n, kemudian mengembalikan nilai ke n dari argumen berikutnya. Hal ini mudah dilakukan kecuali untuk Februari, dimana ELT() harus mengembalikan 29 atau 28 tergantung pada apakah tahun itu tahun kabisat atau bukan. Dengan bahasa API, anda dapat menulis fungsi yang mengembalikan jumlah hari dalam bulan selama tanggal itu terjadi. Versi Perl nya disajikan berikut ini :

```
sub days_in_month
{
my $date = shift;
my $year = substr ($date, 0, 4);
my $month = substr ($date, 5, 2); # month, 1-based
my @days_in_month = (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
my $days = $days_in_month[$month-1];
my $is_leap = ($year % 4 == 0) && ($year % 100 != 0 || $year % 400 == 0);
$days++ if $month == 2 && $is_leap; # add a day for Feb of leap years
return ($days);
}
```

1.21 Memperlakukan Tanggal atau Waktu sebagai Angka Penyelesaian

Lakukan konversi dari string ke angka.

Pembahasan

Dalam banyak kasus, MySQL mampu memperlakukan tanggal dan waktu sebagai angka. Hal ini kadang-kadang berguna jika anda ingin melakukan operasi aritmetika pada nilai tersebut. Untuk memaksakan konversi nilai waktu ke bentuk angka, tambahkan nol atau gunakan angka itu sebagai angka :

```
mysql> SELECT t1, t1+0 AS 't1 as number', FLOOR(t1) AS 't1 as number',
-> FLOOR(t1/10000) AS 'hour part' FROM time_val;
```

t1	t1 as number	t1 as number	hour part
15:00:00	150000	150000	15
05:01:30	50130	50130	5
12:30:20	123020	123020	12

Konversi yang sama dapat dilakukan untuk nilai tanggal atau tanggal dan waktu :

```
mysql> SELECT d, d+0 FROM date_val;
```

d	d+0
1864-02-28	18640228
1900-01-15	19000115
1987-03-05	19870305
1999-12-31	19991231
2000-06-04	20000604

```
mysql> SELECT dt, dt+0 FROM datetime_val;
```

dt	dt+0
1970-01-01 00:00:00	19700101000000
1987-03-05 12:30:15	19870305123015
1999-12-31 09:00:00	19991231090000
2000-06-04 15:45:30	20000604154530

Nilai yang dihasilkan dengan menambahkan nol tidak sama dengan yang dihasilkan dengan konversi ke satuan dasarnya seperti detik atau hari. Hasilnya adalah yang anda dapatkan dengan menghilangkan pembatas dari penyajian string dari nilai asli. Selain itu, konversi ke bentuk angka juga bisa dilakukan dengan MySQL jika nilai yang dikonversi dianggap sebagai nilai waktu oleh MySQL. Jika anda mencoba mengkonversi string ke angka dengan menambah nol, anda akan mendapatkan komponen pertamanya saja :

```
mysql> SELECT '1999-01-01'+0, '1999-01-01 12:30:45'+0, '12:30:45'+0;
```

'1999-01-01'+0	'1999-01-01 12:30:45'+0	'12:30:45'+0
1999	1999	12

Kejadian yang sama juga terjadi dengan fungsi seperti `DATE_FORMAT()` dan `TIME_FORMAT()`, atau jika anda mengambil bagian dari nilai `DATETIME` atau `TIMESTAMP` dengan `LEFT()` atau `RIGHT()`. Dalam kaitannya dengan +0, hasil dari fungsi ini diperlakukan sebagai tipe string dan bukan tipe waktu.

1.22 Memaksa MySQL untuk Memperlakukan String sebagai Nilai Waktu

Penyelesaian

Gunakan string dalam konteks waktu untuk memberi MySQL cara bagaimana untuk memperlakukannya.

Pembahasan

Jika anda ingin MySQL memperlakukan string sebagai tanggal atau waktu, gunakanlah keduanya dalam ekspresi yang menyediakan konteks waktu tanpa merubah nilainya. Sebagai contoh, anda tidak dapat menambah nol ke string TIME supaya bisa dikonversi dari waktu ke angka, tapi jika anda menggunakan `TIME_TO_SEC()` dan `SEC_TO_TIME()` anda dapat :

```
mysql> SELECT SEC_TO_TIME(TIME_TO_SEC('12:30:45'))+0;
+-----+
| SEC_TO_TIME(TIME_TO_SEC('12:30:45'))+0 |
+-----+
| 123045                                |
+-----+
```

Konversi ke dan dari detik menyebabkan nilai itu tidak berubah tapi hasilnya dalam konteks dimana MySQL memperlakukan hasilnya sebagai nilai TIME. Untuk nilai tanggal, prosedurnya mirip, tapi menggunakan `TO_DAYS()` dan `FROM_DAYS()` :

```
mysql> SELECT '1999-01-01'+0, FROM_DAYS(TO_DAYS('1999-01-01'))+0;
+-----+-----+
| '1999-01-01'+0 | FROM_DAYS(TO_DAYS('1999-01-01'))+0 |
+-----+-----+
| 1999           | 19990101                             |
+-----+-----+
```

Untuk string yang berformat DATETIME atau TIMESTAMP, anda dapat menggunakan `DATE_ADD()` untuk memasukkan konteks waktu :

```
mysql> SELECT
-> DATE_ADD('1999-01-01 12:30:45',INTERVAL 0 DAY)+0 AS 'numeric datetime',
-> DATE_ADD('19990101123045',INTERVAL 0 DAY)+0 AS 'numeric timestamp';
+-----+-----+
| numeric datetime | numeric timestamp |
+-----+-----+
| 19990101123045   | 19990101123045    |
+-----+-----+
```

1.23 Memilih Rekord Berdasarkan pada Karakteristik Waktunya

Penyelesaian

Gunakan kondisi tanggal atau waktu dalam klausa WHERE. Ini bisa didasarkan pada perbandingan langsung dari nilai kolom dengan nilai yang diketahui. Atau penting juga untuk menggunakan fungsi untuk nilai kolom untuk mengkonversinya ke bentuk yang lebih cocok untuk pengecekan, seperti penggunaan `MONTH()` untuk mengecek bagian bulan dari tanggal.

Pembahasan

Sebagian besar teknik yang berdasarkan tanggal yang dijelaskan dimuka digambarkan dengan contoh query yang menghasilkan nilai tanggal atau waktu sebagai keluaran. Anda dapat menggunakan teknik yang sama dalam klausa WHERE untuk menempatkan batasan berdasarkan tanggal pada rekord yang dipilih oleh sebuah query. Sebagai contoh, anda dapat memilih rekord yang terjadi sebelum atau setelah suatu tanggal, dalam suatu jangkauan tanggal, atau yang cocok dengan nilai bulan atau hari tertentu.

Membandingkan Tanggal dengan Tanggal

Query berikut ini mencari rekord dari table `date_val` yang terjadi baik sebelum 1900 atau selama tahun 1900 :


```
mysql> SELECT d FROM date_val where d < '1900-01-01';
```

```
+-----+
| d      |
+-----+
| 1864-02-28 |
+-----+
```

```
mysql> SELECT d FROM date_val where d BETWEEN '1900-01-01' AND '1999-12-31';
```

```
+-----+
| d      |
+-----+
| 1900-01-15 |
| 1987-03-05 |
| 1999-12-31 |
+-----+
```

Jika versi MySQL anda lebih tua dari 3.23.9, salah satu masalah untuk mengecek adalah bahwa BETWEEN kadang-kadang tidak bekerja dengan benar dengan string tanggal jika mereka tidak dalam format ISO. Misalnya, contoh ini salah :

```
SELECT d FROM date_val WHERE d BETWEEN '1960-3-1' AND '1960-3-15';
```

Jika hal ini terjadi, cobalah menulis ulang tanggalnya dalam format ISO untuk mendapatkan hasil yang lebih baik :

```
SELECT d FROM date_val WHERE d BETWEEN '1960-03-01' AND '1960-03-15';
```

Anda dapat juga menulis ulang ekspresi menggunakan dua perbandingan secara eksplisit :

```
SELECT d FROM date_val WHERE d >= '1960-03-01' AND d <= '1960-03-15';
```

Jika anda tidak tahu tanggalnya dengan pasti yang anda inginkan untuk klausa WHERE, anda kadang-kadang dapat menghitungnya menggunakan sebuah ekspresi. Misalnya untuk melaksanakan query “hari ini dalam sejarah” untuk mencari rekord dalam table history untuk mencari kejadian-kejadian yang terjadi tepat 50 tahun yang lalu, lakukan ini :

```
SELECT * FROM history WHERE d = DATE_SUB(CURDATE( ),INTERVAL 50 YEAR);
```

Anda melihat hal semacam in pada surat kabar yang menjalankan kolom yang menunjukkan berita kejadian-kejadian dimasa lalu. Jika and ingin mengambil kejadian yang terjadi “*hari ini*” pada sembarang tahun dan bukannya “*pada tanggal ini*” untuk tahun tertentu, querynya sedikit berbeda. Dalam kasus ini anda harus mencari rekord yang cocok dengan hari kalender sekarang, tanpa melihat tahunnya.

Tanggal yang dihitung berguna untuk pengecekan jangkauan juga. Misalnya, untuk mencari tanggal yang terjadi pada enam tahun yang lalu, gunakan DATE_SUB() untuk menghitung potongan tanggal :

```
mysql> SELECT d FROM date_val WHERE d >= DATE_SUB(CURDATE( ),INTERVAL 6 YEAR);
```

```
+-----+
| d      |
+-----+
| 1999-12-31 |
+-----+
```

| 2000-06-04 |
+-----+

Membandingkan Waktu dengan Waktu

Perbandingan yang melibatkan waktu mirip dengan yang melibatkan tanggal. Sebagai contoh, untuk mencari waktu yang terjadi dari 9 AM hingga 2 PM, gunakan ekspresi seperti berikut ini :

```
... WHERE t1 BETWEEN '09:00:00' AND '14:00:00';  
... WHERE HOUR(t1) BETWEEN 9 AND 14;
```

Untuk kolom TIME yang diindeks, metoda pertama akan lebih efisien. Metoda kedua mempunyai sifat dimana dia dapat bekerja tidak hanya untuk kolom TIME tapi juga untuk kolom DATETIME dan TIMESTAMP juga.

Membandingkan Tanggal dengan Hari Kalender

Untuk menjawab pertanyaan tentang hari tertentu dalam suatu tahun, gunakan pengecekan hari kalender. Contoh berikut ini menggambarkan bagaimana untuk melakukannya dalam konteks untuk mencari hari ulang tahun :

- . Siapa yang berulang tahun hari ini ? Ini membutuhkan pencocokan hari kalender tertentu, sehingga anda mengambil bulan dan tanggal tanpa memperhatikan tahun ketika melakukan perbandingan :

```
... WHERE MONTH(d) = MONTH(CURDATE( )) AND DAYOFMONTH(d) =  
      DAYOFMONTH(CURDATE( ));
```

Query semacam ini biasanya diaplikasikan dalam data biografi untuk mencari daftar aktor, politikus, musikus, dan sebagainya, yang lahir pada hari tertentu dalam suatu tahun. Anda bisa menggunakan DAYOFYEAR() untuk menyelesaikan masalah “pada hari ini”, karena ini akan menghasilkan query yang lebih sederhana. Tapi DAYOFYEAR() tidak bekerja dengan sempurna untuk tahun kabisat. Kehadiran 29 Februari akan menghilangkan nilai hari dari Maret hingga Desember.

- . Siapa yang berulang tahun bulan ini? Dalam kasus ini, perlu untuk mengecek hanya bulannya saja :

```
... WHERE MONTH(d) = MONTH(CURDATE( ));
```

- . Siapa yang berulang tahun bulan depan ? Yang perlu diperhatikan disini adalah anda tidak dapat hanya menambahkan satu ke bulan ini untuk mendapatkan nomor bulan yang memenuhi tanggal-tanggal yang cocok. Hal ini akan memberikan hasil 13 untuk tanggal pada bulan Desember. Untuk memastikan anda mendapatkan 1 Januari, gunakan kedua teknik ini :

```
.. WHERE MONTH(d) = MONTH( DATE_ADD( CURDATE( ), INTERVAL 1 MONTH ) );  
.. WHERE MONTH(d) = MOD( MONTH( CURDATE( ) ), 12 ) + 1;
```

1.24 Menggunakan Nilai TIMESTAMP

Penyelesaian

Tipe kolom TIMESTAMP dapat digunakan untuk hal ini. Bagaimanapun juga, tipe kolom ini mempunyai sifat yang kadang-kadang mengejutkan, sehingga bacalah bagian ini untuk memastikan anda tahu apa yang akan anda dapatkan.

Pembahasan

MySQL mendukung tipe kolom `TIMESTAMP` yang dalam banyak hal dapat diperlakukan sama seperti tipe `DATETIME`. Tapi tipe `TIMESTAMP` mempunyai sifat khusus :

- Kolom `TIMESTAMP` pertama dalam table adalah khusus pada saat pembuatan rekord: nilai defaultnya adalah tanggal dan waktu sekarang. Ini berarti anda tidak perlu menyatakan nilainya sama sekali dalam pernyataan `INSERT` jika anda ingin kolom ini mengeset ke waktu pembentukan rekord. MySQL akan membuatnya secara otomatis.
- `TIMESTAMP` pertama juga khusus kapanpun sembarang kolom dalam baris dirubah dari nilai sekarang. MySQL secara otomatis mengupdate nilainya ke tanggal dan waktu pada saat perubahan dibuat.
- Kolom `TIMESTAMP` lainnya dalam table tidak khusus seperti yang pertama. Nilai defaultnya adalah nol, bukan tanggal dan waktu sekarang. Juga nilainya tidak berubah secara otomatis ketika anda merubah kolom lainnya. Untuk merubahnya, anda harus merubahnya sendiri.
- Kolom `TIMESTAMP` dapat diset ke tanggal dan waktu sekarang pada setiap saat dengan menyetelnya ke `NULL`. Ini bekerja untuk semua kolom `TIMESTAMP`, tidak hanya yang pertama. Sifat `TIMESTAMP` yang berhubungan dengan pembuatan rekord dan modifikasi membuat tipe kolom ini cocok untuk tipe masalah tertentu.

1.25 Merekam Waktu Modifikasi Terakhir dari sebuah Baris

Penyelesaian

Masukkan kolom `TIMESTAMP` dalam table anda.

Pembahasan

Untuk membuat sebuah table dimana masing-masing baris mengandung nilai yang menunjukkan rekord yang diupdate paling akhir, masukkan kolom `TIMESTAMP`. Kolom ini akan mengeset ke tanggal dan waktu sekarang ketika anda membuat baris baru, dan diperbarui kapanpun anda mempebarui nilai dari kolom lain pada baris itu. Misalkan anda membuat sebuah table `tsdemo1` dengan kolom `TIMESTAMP` yang terlihat seperti ini :

```
CREATE TABLE tsdemo1
(
  t TIMESTAMP,
  val INT
);
```

Masukkan beberapa rekord kedalam table dan kemudian pilihlah isinya. Pernyataan `INSERT` pertama menunjukkan bahwa anda dapat mengeset `t` ke tanggal dan waktu sekarang dengan mengesetnya secara eksplisit ke `NULL`. Pernyataan kedua menunjukkan bahwa anda mengeset `t` dengan menghilangkannya dari pernyataan `INSERT` :

```
mysql> INSERT INTO tsdemo1 (t,val) VALUES(NULL,5);
mysql> INSERT INTO tsdemo1 (val) VALUES(10);
mysql> SELECT * FROM tsdemo1;
```

```
+-----+-----+
| t           | val |
+-----+-----+
| 20020715115825 | 5   |
```

t	val
20020715115831	10

Berikut adalah query yang merubah satu record kolom val dan mengecek efeknya pada isi table :

```
mysql> UPDATE tsdemo1 SET val = 6 WHERE val = 5;
mysql> SELECT * FROM tsdemo1;
```

t	val
20020715115915	6
20020715115831	10

Hasilnya menunjukkan bahwa TIMESTAMP telah dirubah hanya untuk record yang dimodifikasi. Jika anda memodifikasi banyak record, nilai TIMESTAMP seluruhnya akan dirubah :

```
mysql> UPDATE tsdemo1 SET val = val + 1;
mysql> SELECT * FROM tsdemo1;
```

t	val
20020715115926	7
20020715115926	11

Mengeluarkan pernyataan UPDATE yang sebenarnya tidak merubah nilai pada kolom val tidak mengupdate nilai TIMESTAMP. Untuk melihat hal ini, set setiap record dalam kolom val ke nilai sekarang, kemudian review isi table :

```
mysql> UPDATE tsdemo1 SET val = val + 0;
mysql> SELECT * FROM tsdemo1;
```

t	val
20020715115926	7
20020715115926	11

Sebuah alternatif menggunakan TIMESTAMP adalah menggunakan kolom DATETIME dan set ke NOW() secara eksplisit ketika anda membuat sebuah record dan kapanpun anda mengupdate sebuah record. Dalam kasus ini semua penggunaan yang menggunakan table itu harus menggunakan strategi yang sama, yang akan gagal bahkan jika satu aplikasi menolaknya.

1.26 Merekam Waktu Pembuatan Baris

Penyelesaian

Anda hanya perlu memasukkan kolom TIMESTAMP kedua yang mempunyai sifat berbeda dengan yang pertama.

Pembahasan

Jika anda ingin sebuah kolom diset mula-mula ke waktu pada saat sebuah record dibuat, tapi tetap konstan, sebuah TIMESTAMP bukanlah jawabannya, karena ini akan diupdate kapanpun kolom lainnya dalam record diupdate. Dalam hal ini harus digunakan dua kolom TIMESTAMP dan mengambil keuntungan dari fakta bahwa TIMESTAMP kedua tidak akan mempunyai sifat khusus

yang sama dari yang pertama. Kedua kolom dapat di set ke tanggal dan waktu sekarang ketika record dibuat

```
CREATE TABLE tsdemo2
(
  t_update TIMESTAMP, # record last-modification time
  t_create TIMESTAMP, # record creation time
  val INT
);
```

Buatlah table, kemudian masukkan kedalamnya record-record berikut ini yang kedua kolom TIMESTAMP nya diset ke NULL, untuk menginisialisasinya ke tanggal dan waktu sekarang :

```
mysql> INSERT INTO tsdemo2 (t_update,t_create,val) VALUES(NULL,NULL,5);
mysql> SELECT * FROM tsdemo2;
```

t_update	t_create	val
20020715120003	20020715120003	5

Setelah memasukkan record, rubahlah kolom val, kemudian ceklah bahwa update memodifikasi kolom t_update dan membiarkan kolom t_create diset ke waktu pembuatan record :

```
mysql> UPDATE tsdemo2 SET val = val + 1;
mysql> SELECT * FROM tsdemo2;
```

t_update	t_create	val
20020715120012	20020715120003	6

Seperti pada table tsdemo1, updatelah record-record tsdemo2 yang sebenarnya tidak memodifikasi sebuah kolom tidak menyebabkan perubahan nilai TIMESTAMP sama sekali :

```
mysql> UPDATE tsdemo2 SET val = val + 0;
mysql> SELECT * FROM tsdemo2;
```

t_update	t_create	val
20020715120012	20020715120003	6

Strategi alternatif adalah menggunakan kolom DATETIME untuk t_create dan t_update. Ketika membuat sebuah record, set keduanya ke NOW() secara eksplisit. Ketika memodifikasi sebuah record, updatelah t_update ke NOW() dan biarkan t_create sendiri.

1.27 Melakukan Perhitungan dengan Nilai TIMESTAMP

Penyelesaian

Nilai TIMESTAMP dapat dipakai dalam perhitungan seperti juga nilai DATETIME, seperti perbandingan, pergeseran dan pengambilan komponen.

Pembahasan

Query berikut ini menunjukkan beberapa operasi yang mungkin yang dapat anda lakukan pada nilai `TIMESTAMP` menggunakan table `tsdemo2`.

- . Rekords yang belum dimodifikasi sejak pembuatannya :
`SELECT * FROM tsdemo2 WHERE t_create = t_update;`
- . Rekords yang dimodifikasi dalam 12 jam terakhir :
`SELECT * FROM tsdemo2 WHERE t_update >= DATE_SUB(NOW(), INTERVAL 12 HOUR);`
- . Perbedaan antara waktu pembuatan dan waktu (*ditunjukkan dalam detik dan jam*)
`SELECT t_create, t_update,
UNIX_TIMESTAMP(t_update) - UNIX_TIMESTAMP(t_create) AS 'seconds',
(UNIX_TIMESTAMP(t_update) - UNIX_TIMESTAMP(t_create))/(60 * 60) AS
hours'FROM tsdemo2;`
- . Rekords yang dibuat dari 1 PM hingga 4 PM:
`SELECT * FROM tsdemo2
WHERE HOUR(t_create) BETWEEN 13 AND 16;
atau
SELECT * FROM tsdemo2
WHERE DATE_FORMAT(t_create,'%H%i%s') BETWEEN '130000' AND '160000';
Or even by using TIME_TO_SEC() to strip off the date part of the t_create values:
SELECT * FROM tsdemo2
WHERE TIME_TO_SEC(t_create)
BETWEEN TIME_TO_SEC('13:00:00') AND TIME_TO_SEC('16:00:00');`

1.28 Menampilkan Nilai `TIMESTAMP` dalam Bentuk yang dapat Dibaca

Penyelesaian

Format ulang nilai tersebut dengan fungsi `DATE_FORMAT()`.

Pembahasan

Kolom `TIMESTAMP` mempunyai sifat khusus yang diinginkan, tapi salah satu yang kadang-kadang tidak begitu diinginkan adalah format tampilan (`CCYYMMDDhhmmss`). Sebagai string digit yang tidak terpotong, ini tidak konsisten dengan format `DATETIME` (`CCYY-MM-DD hh:mm:ss`) dan juga lebih sulit untuk dibaca. Untuk menulis ulang nilai `TIMESTAMP` kedalam format `DATETIME`, gunakan fungsi `DATE_FORMAT()`. Contoh berikut ini menggunakan tabel `tsdemo2`.

```
mysql> SELECT t_create, DATE_FORMAT(t_create,'%Y-%m-%d %T') FROM tsdemo2;  
+-----+-----+  
| t_create          | DATE_FORMAT(t_create,'%Y-%m-%d %T') |  
+-----+-----+  
| 20020715120003    | 2002-07-15 12:00:03                 |  
+-----+-----+
```

Anda dapat melakukan dari arah sebaliknya, meskipun hal ini tidak umum. Salah satu cara adalah menggunakan `DATE_FORMAT()`, lainnya adalah dengan menambah nol.

```
mysql> SELECT dt, DATE_FORMAT(dt,'%Y%m%d%H%i%s'), dt+0  
-> FROM datetime_val;  
+-----+-----+-----+-----+
```

dt	DATE_FORMAT(dt, '%Y%m%d%H%i%s')	dt+0
1970-01-01 00:00:00	19700101000000	19700101000000
1987-03-05 12:30:15	19870305123015	19870305123015
1999-12-31 09:00:00	19991231090000	19991231090000
2000-06-04 15:45:30	20000604154530	20000604154530

Referensi:

Paul DuBois, *“MySQL Cookbook”*, O'Reilly, October 2002

BIOGRAFI PENULIS



Janner Simarmata. Lahir di Aek Nabara, 07 Januari 1976. Tamat dari STM GKPS Pematang Siantar tahun 1995. Menyelesaikan program S1 pada jurusan Teknik Informatika di STMIK BANDUNG pada tahun 2000. Pernah mengajar di beberapa Perguruan Tinggi Swasta seperti: STMIK Mikroskil, STMIK Multimedia Prima, Unika Santo Thomas Sumatera Utara. Pada tahun 2004 melanjutkan studi pada program S2 (M.Kom) pada jurusan Ilmu Komputer Universitas Gadjadara sampai sekarang.

Informasi lebih lanjut tentang penulis:

KEYWORD: *Janner Simarmata*

Email: *sijanner@yahoo.com*