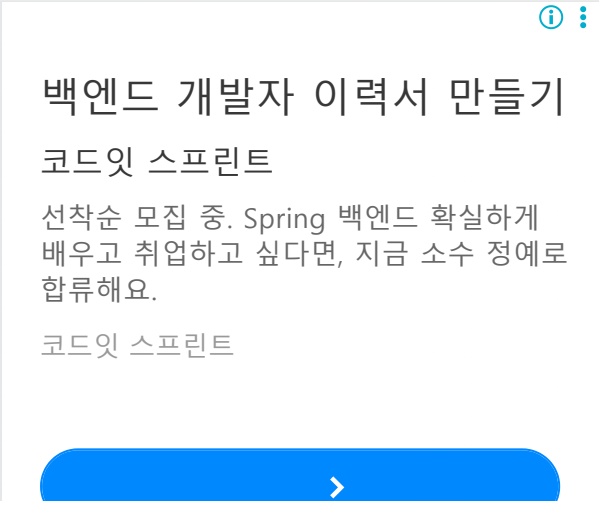
MySQL TIME Data Type



Summary: in this tutorial, you will learn about the MySQL `TIME` data type and how to use temporal functions to manipulate time data effectively.

Introduction to MySQL TIME data type

MySQL uses the `'HH:MM:SS'` format for querying and displaying a time value that represents a time of day, which is within 24 hours.

To represent a time interval between two events, MySQL uses the `'HHH:MM:SS'` format, which is larger than 24 hours.

To define a `TIME` column, you use the following syntax:

```
column_name TIME;
```

For example, the following snippet defines a column named `start_at` with `TIME` data type.

```
start_at TIME;
```

A `TIME` value ranges from `-838:59:59` to `838:59:59`. In addition, a `TIME` value can have a fractional seconds part that is up to microseconds precision (6 digits).

To define a column whose data type is `TIME` with a fractional-second precision part, you use the following syntax:

```
column_name TIME(N);
```

In this syntax, `N` is an integer that represents the fractional part, which is up to 6 digits.

The following defines a column with `TIME` data type including 3 digits of fractional seconds.

```
begin_at TIME(3);
```

A `TIME` value takes 3 bytes for storage. If a `TIME` value includes fractional second precision, it will take additional bytes based on the number of digits of the fractional second precision.

The following table illustrates the storage required for fractional second precision.

Fractional Second Precision	Storage (BYTES)
0	0
1, 2	1
3, 4	2
5, 6	3

For example, `TIME` and `TIME(0)` takes 3 bytes. `TIME(1)` and `TIME(2)` takes 4 bytes (3 + 1); `TIME(3)` and `TIME(6)` take 5 and 6 bytes.

MySQL TIME data type example

Let's take a look at an example of using the `TIME` data type for columns in a table.

First, create a new table named `tests` that consists of four columns: `id`, `name`, `start_at`, and `end_at`. The data types of the `start_at` and `end_at` columns are `TIME`:

```
CREATE TABLE tests (
    id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    name VARCHAR(255) NOT NULL,
    start_at TIME,
    end_at TIME
);
```

Second, insert a row into the `tests` table.

```
INSERT INTO tests(name,start_at,end_at)
VALUES('Test 1', '08:00:00','10:00:00');
```

Third, query data from the `tests` table.

```
SELECT
    name, start_at, end_at
FROM
    tests;
```

Output:

```
+--------+----------+----------+
| name   | start_at | end_at   |
+--------+----------+----------+
| Test 1 | 08:00:00 | 10:00:00 |
+--------+----------+----------+
1 row in set (0.00 sec)
```

Notice that we use `'HH:MM:SS'` as the literal time value in the `INSERT` statement. Let's examine all the valid time literals that MySQL can recognize.

MySQL TIME literals

MySQL recognizes various time formats besides the `'HH:MM:SS'` format that we mentioned earlier.

MySQL allows you to use the `'HHMMSS'` format without delimiter (:) to represent time value. For example, `'08:30:00'` and `'10:15:00'` can be rewritten as `'083000'` and `'101500'`.

```
INSERT INTO tests(name,start_at,end_at)
VALUES('Test 2','083000','101500');
```

The tests table will have the following rows:

```
+--------+----------+----------+
| name   | start_at | end_at   |
+--------+----------+----------+
| Test 1 | 08:00:00 | 10:00:00 |
| Test 2 | 08:30:00 | 10:15:00 |
+--------+----------+----------+
2 rows in set (0.00 sec)
```

However, `108000` is not a valid time value because `80` does not represent the correct minute. In this case, MySQL will raise an error if you try to insert an invalid time value into a table.

```
INSERT INTO tests(name,start_at,end_at)
VALUES('Test invalid','083000','108000');
```

MySQL issued the following error message after executing the above statement.

```
Error Code: 1292. Incorrect time value: '108000' for column 'end_at' at row 1
```

In addition to the string format, MySQL accepts the `HHMMSS` as a number that represents a time value. You can also use `SS` , `MMSS` . For example, instead of using `'082000'` , you can use `082000` as follows:

```
INSERT INTO tests(name,start_at,end_at)
VALUES('Test 3',082000,102000);
```

```
+----+--------+----------+----------+
| id | name   | start_at | end_at   |
+----+--------+----------+----------+
|  1 | Test 1 | 08:00:00 | 10:00:00 |
|  2 | Test 2 | 08:30:00 | 10:15:00 |
|  3 | Test 3 | 08:20:00 | 10:20:00 |
+----+--------+----------+----------+
```

```
3 rows in set (0.00 sec)
```

For the time interval, you can use the `'D HH:MM:SS'` format where `D` represents days with a range from 0 to 34. A more flexible syntax is `'HH:MM'`, `'D HH:MM'`, `'D HH'`, or `'SS'`.

If you use the delimiter `:`, you can use one digit to represent hours, minutes, or seconds. For example, `9:5:0` can be used instead of `'09:05:00'`.

```
INSERT INTO tests(name,start_at,end_at)
VALUES('Test 4','9:5:0',100500);
```

```
+----+--------+----------+----------+
| id | name   | start_at | end_at   |
+----+--------+----------+----------+
|  1 | Test 1 | 08:00:00 | 10:00:00 |
|  2 | Test 2 | 08:30:00 | 10:15:00 |
|  3 | Test 3 | 08:20:00 | 10:20:00 |
|  4 | Test 4 | 09:05:00 | 10:05:00 |
+----+--------+----------+----------+
4 rows in set (0.00 sec)
```

Useful MySQL TIME functions

MySQL provides several useful temporal functions for manipulating `TIME` data.

1) Getting the current time

To get the current time of the database server, you use the `CURRENT_TIME` function. The `CURRENT_TIME` function returns the current time value as a string (`'HH:MM:SS'`) or a numeric value (`HHMMSS`) depending on the context where the function is used.

The following statements illustrate the `CURRENT_TIME` function in both string and numeric contexts:

```
SELECT
    CURRENT_TIME() AS string_now,
    CURRENT_TIME() + 0 AS numeric_now;
```

2) Adding and Subtracting time from a TIME value

To add a `TIME` value to another `TIME` value, you use the `ADDTIME` function. To subtract a `TIME` value from another `TIME` value, you use the `SUBTIME` function.

The following statement adds and subtracts 2 hours 30 minutes to and from the current time.

```
SELECT
    CURRENT_TIME(),
    ADDTIME(CURRENT_TIME(), 023000),
    SUBTIME(CURRENT_TIME(), 023000);
```

```
+---------------+---------------------------------+---------------------------------+
| CURRENT_TIME() | ADDTIME(CURRENT_TIME(), 023000) | SUBTIME(CURRENT_TIME(), 023000) |
+---------------+---------------------------------+---------------------------------+
| 16:23:31       | 18:53:31                        | 13:53:31                        |
+---------------+---------------------------------+---------------------------------+
1 row in set (0.00 sec)
```

In addition, you can use the `TIMEDIFF()` function to get a difference between two `TIME` values.

```
SELECT
    TIMEDIFF(end_at, start_at)
FROM
    tests;
```

3) Formatting MySQL TIME values

Although MySQL uses `'HH:MM:SS'` when retrieving and displaying a `TIME` value, you can display the `TIME` value in your preferred way using the `TIME_FORMAT` function.

The `TIME_FORMAT` function is like the `DATE_FORMAT` function except that the `TIME_FORMAT` function is used to format a `TIME` value only.

See the following example.

```
SELECT
```

```
    name,
    TIME_FORMAT(start_at, '%h:%i %p') start_at,
    TIME_FORMAT(end_at, '%h:%i %p') end_at
FROM
    tests;
```

Output:

```
+--------+----------+----------+
| name   | start_at | end_at   |
+--------+----------+----------+
| Test 1 | 08:00 AM | 10:00 AM |
| Test 2 | 08:30 AM | 10:15 AM |
| Test 3 | 08:20 AM | 10:20 AM |
| Test 4 | 09:05 AM | 10:05 AM |
+--------+----------+----------+
4 rows in set (0.00 sec)
```

In the time format string above:

- `%h` means two-digit hours from 0 to 12.
- `%i` means two-digit minutes from 0 to 60.
- `%p` means AM or PM.

4) Extracting hour, minute, and second from a TIME value

To extract the hour, minute, and second from a `TIME` value, you use `HOUR`, `MINUTE`, and `SECOND` functions as follows:

```
SELECT
  start_at,
  hour(start_at) hour,
  minute(start_at) minute,
  second(start_at) second
FROM
  tests;
```

Output:

```
+----------+------+--------+--------+
| start_at | hour | minute | second |
+----------+------+--------+--------+
| 08:00:00 |    8 |      0 |      0 |
| 08:30:00 |    8 |     30 |      0 |
| 08:20:00 |    8 |     20 |      0 |
| 09:05:00 |    9 |      5 |      0 |
+----------+------+--------+--------+
4 rows in set (0.00 sec)
```

5) Getting UTC time value

To get the UTC time, you use `UTC_TIME` function as follows:

```
SELECT
   CURRENT_TIME(),
   UTC_TIME();
```

Output:

```
+----------------+------------+
| CURRENT_TIME() | UTC_TIME() |
+----------------+------------+
| 16:24:43       | 09:24:43   |
+----------------+------------+
1 row in set (0.00 sec)
```

In this tutorial, we have been covered a lot about MySQL `TIME` data type and some commonly used temporal functions for manipulating `TIME` values.

Was this tutorial helpful?

ADVERTISEMENTS

PREVIOUSLY
MySQL DATE Data Type

UP NEXT
MySQL CHAR Data Type

ADVERTISEMENTS

Search ...

GETTING STARTED

What Is MySQL?

Install MySQL Database Server

Connect to MySQL Server

Download MySQL Sample Database

Load Sample Database

QUERYING DATA

SELECT FROM

SELECT

ORDER BY

WHERE

SELECT DISTINCT

AND

OR

IN

NOT IN

BETWEEN

LIKE

LIMIT

IS NULL

Table & Column Aliases

Joins

INNER JOIN

LEFT JOIN

RIGHT JOIN

Self Join

CROSS JOIN

GROUP BY

HAVING

HAVING COUNT

ROLLUP

Subquery

Derived Tables

EXISTS

EXCEPT

INTERSECT

ADVERTISEMENTS

MANAGING DATABASES

Select a Database

Create Databases

Drop Databases

MANAGING TABLES

Create Tables

AUTO_INCREMENT

Rename Tables

Add Columns

Drop Columns

Drop Tables

Temporary Tables

Generated Columns

MYSQL CONSTRAINTS

Primary Key

Foreign Key

Disable Foreign Key Checks

UNIQUE Constraint

NOT NULL Constraint

DEFAULT Constraint

CHECK Constraint

ADVERTISEMENTS

INSERT DATA

Insert Into

Insert Multiple Rows

INSERT INTO SELECT

Insert On Duplicate Key Update

INSERT IGNORE

Insert DateTimes

Insert Dates

UPDATE DATA

UPDATE

UPDATE JOIN

DELETE DATA

DELETE JOIN

ON DELETE CASCADE

TRUNCATE TABLE

MYSQL TRANSACTIONS

Table Locking

MYSQL DATA TYPES

BIT

INT

BOOLEAN

DECIMAL

DATETIME

TIMESTAMP

DATE

TIME

CHAR

VARCHAR

TEXT

BINARY

VARBINARY

ENUM

BLOB

MYSQL GLOBALIZATION

MySQL Character Sets

MySQL Collation

MYSQL IMPORT & EXPORT

Import a CSV File Into a Table

Export a Table to a CSV File

ADVERTISEMENTS

innodb_log_buffer_size: Configure InnoDB Log Buffer Size

innodb_buffer_pool_chunk_size: Configure Buffer Pool Chunk Size

innodb_buffer_pool_instances: Configuring Multiple Buffer Pool Instances for Improved Concurrency in MySQL

innodb_buffer_pool_size: Configure InnoDB Buffer Pool Size

MySQL InnoDB Architecture

How to Kill a Process in MySQL

MySQL Resources

MySQL Books

Copyright © 2008 - Present by www.mysqltutorial.org. All Rights Reserved.