Home » MySQL Basics » MySQL BIT Data Type

MySQL BIT Data Type

Summary: in this tutorial, you will learn about MySQL `BIT` data type and how to store `BIT` data in a column of a table.

Introduction to MySQL BIT data type

The `BIT` data type that allows you to store bit values, which are 0 and 1.

Here's the syntax for defining `BIT` type of a column:

```
column_name BIT(n)
```

The `BIT(n)` can store up to n-bit values. The `n` can range from 1 to 64. The default value of n is 1 if you skip it.

So the following syntaxes are equivalent:

```
column_name BIT(1);
```

and

```
column_name BIT;
```

BIT literals

To specify a bit value literal, you use `b'val'` or `0bval` notation, which `val` is a binary value that contains only 0 and 1.

The leading `b` can be written as `B`, for example, the following are valid bit literals:

```
b01
B11
```

However, the leading `0b` is case-sensitive. Therefore, you cannot use `0B`. For example, the following is an invalid bit literal value:

```
0B'1000'
```

By default the character set of a bit-value literal is the binary string as follows:

```
SELECT CHARSET(B'); -- binary
```

MySQL BIT data type examples

The following statement creates a new table named `working_calendars` that has the days column is `BIT(7)`:

```
CREATE TABLE working_calendars(
    year INT,
    week INT,
    days BIT(7),
    PRIMARY KEY(year,week)
);
```

The values in the column `days` indicate whether the day is a working day or day off i.e., 1: working day and 0: day off.

Suppose that Saturday and Friday of the first week of 2017 are not working days, you can insert a row into the `working_calendars` table:

```
INSERT INTO working_calendars(year,week,days)
VALUES(2017,1,B'1111100');
```

The following query retrieves data from the `working_calendar` table:

```
SELECT
    year, week, days
FROM
    working_calendars;
```

Output:

```
+------+------+------------+
| year | week | days       |
+------+------+------------+
| 2017 |    1 | 0x7C       |
+------+------+------------+
1 row in set (0.00 sec)
```

The output indicates that the bit value in the `days` column is converted into an integer. To represent it as bit values, you use the `BIN` function:

```
SELECT
    year, week, bin(days)
FROM
    working_calendars;
```

Output:

```
+------+------+-----------+
| year | week | bin(days) |
+------+------+-----------+
| 2017 |    1 | 1111100   |
+------+------+-----------+
1 row in set (0.00 sec)
```

If you insert a value into a `BIT(n)` column that is less than `n` bits long, MySQL will pad zeros on the left of the bit value.

Suppose the first day of the second week is off, you can insert `01111100` into the `days` column. However, the `111100` value will also work because MySQL will pad one zero to the left.

```
INSERT INTO working_calendars(year,week,days)
VALUES(2017,2,B'111100');
```

To view the data you use the same query as above:

```
SELECT
    year, week , bin(days)
FROM
    working_calendars;
```

Output:

```
+------+------+-----------+
| year | week | bin(days) |
+------+------+-----------+
| 2017 |    1 | 1111100   |
| 2017 |    2 | 111100    |
+------+------+-----------+
2 rows in set (0.00 sec)
```

The output shows that MySQL removed the leading zeros before returning the result. To display it correctly, you can use the `LPAD()` function:

```
SELECT
    year, week, lpad(bin(days),7,'0')
FROM
    working_calendars;
```

Output:

```
+------+------+----------------------+
| year | week | lpad(bin(days),7,'0') |
+------+------+----------------------+
| 2017 |    1 | 1111100              |
| 2017 |    2 | 0111100              |
+------+------+----------------------+
2 rows in set (0.00 sec)
```

The output shows the expected format.

Summary

- Use MySQL `BIT` data type to store `BIT` data in a table.

Was this tutorial helpful? 👍 👎

PREVIOUSLY
MySQL Table Locking

UP NEXT
MySQL INT Data Type

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

MANAGING DATABASES

MANAGING TABLES

Temporary Tables

Generated Columns

MYSQL CONSTRAINTS

Primary Key

Foreign Key

Disable Foreign Key Checks

UNIQUE Constraint

NOT NULL Constraint

DEFAULT Constraint

CHECK Constraint

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

ABOUT MYSQL TUTORIAL WEBSITE

MySQLTutorial.org helps you master MySQL quickly, easily, and with enjoyment. Our tutorials make learning MySQL a breeze.

All MySQL tutorials are clear, practical and easy-to-follow. More About Us

LATEST TUTORIALS

MySQL Port

MySQL Commands

innodb_dedicated_server: Configure InnoDB Dedicated Server

innodb_flush_method: Configure InnoDB Flush Method

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

SITE LINKS

Donation ❤

Contact Us

About

Privacy Policy

OTHERS

MySQL Cheat Sheet

MySQL Resources

MySQL Books