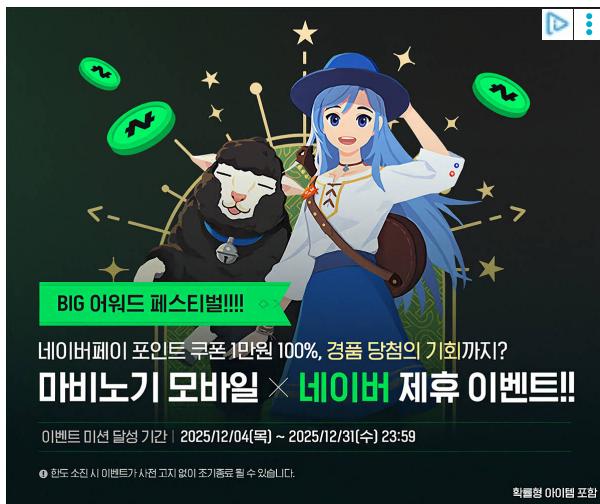


MySQL NOT NULL Constraint



Summary: in this tutorial, you will learn about MySQL `NOT NULL` constraints including defining a `NOT NULL` constraint for a column, adding a `NOT NULL` constraint to an existing column, and removing a `NOT NULL` constraint from a column.

Introduction to MySQL NOT NULL constraints

A `NOT NULL` constraint ensures that values stored in a column are not `NULL`. The syntax for defining a `NOT NULL` constraint is as follows:

```
column_name data_type NOT NULL;
```

A column may have only one `NOT NULL` constraint, which enforces the rule that the column must not contain any `NULL` values.

In other words, if you attempt to [update](#) or [insert](#) a `NULL` value into a `NOT NULL` column, MySQL will issue an error.

For example, the following creates the `tasks` table using the `CREATE TABLE` statement:

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

```
title VARCHAR(255) NOT NULL,  
start_date DATE NOT NULL,  
end_date DATE  
);
```

In the `tasks` table, we explicitly define the `title` and `start_date` columns with `NOT NULL` constraints.

The `id` column has the PRIMARY KEY constraint, therefore, it implicitly includes a `NOT NULL` constraint.

The `end_date` column can have `NULL` values, as when creating a new task, you may not know its completion date

The following shows the structure of the tasks table:

```
DESC tasks;
```

Output:

Field	Type	Null	Key	Default	Extra
<code>id</code>	<code>int</code>	<code>NO</code>	<code>PRI</code>	<code>NULL</code>	<code>auto_increment</code>
<code>title</code>	<code>varchar(255)</code>	<code>NO</code>		<code>NULL</code>	
<code>start_date</code>	<code>date</code>	<code>NO</code>		<code>NULL</code>	
<code>end_date</code>	<code>date</code>	<code>YES</code>		<code>NULL</code>	

4 rows in set (0.01 sec)

It's a good practice to have the `NOT NULL` constraint in every column of a table unless you have a specific reason not to.

Generally, `NULL` values may complicate your queries because you need to use `NONE`-related functions such as [`ISNULL\(\)`](#), [`IFNULL\(\)`](#), and [`NULLIF\(\)`](#) to handle them.

Adding a NOT NULL constraint to an existing column

Typically, you add `NOT NULL` constraints to columns when you create the table. However, you

may want to add a `NOT NULL` constraint to a column of an existing table. In this case, you use the following steps:

1. First, check the current values of the column if there are any `NULL` values.
2. Second, update the `NULL` to non-`NULL`.
3. Third, modify the column with a `NOT NULL` constraint.

Consider the following example.

First, insert some rows into the `tasks` table:

```
INSERT INTO tasks(title ,start_date, end_date)
VALUES('Learn MySQL NOT NULL constraint', '2017-02-01','2017-02-02'),
      ('Check and update NOT NULL constraint to your database', '2017-02-01',NULL);
```

If you want to require users to provide an estimated end date when creating a new task, you can add a `NOT NULL` constraint to the `end_date` column of the `tasks` table.

Second, find rows with `NULLs` in the column `end_date` using the `IS NULL` operator:

```
SELECT * FROM tasks
WHERE end_date IS NULL;
```

Output:

```
+-----+-----+
| id | title                                | start_date | end_date |
+-----+-----+
|  2 | Check and update NOT NULL constraint to your database | 2017-02-01 | NULL     |
+-----+-----+
1 row in set (0.00 sec)
```

The query returned one row with `NULL` in the column `end_date`.

Third, update the `NULL` values to non-null values. In this case, you can create a rule that sets to one week after the start date when the `end_date` is `NULL`.

```
UPDATE tasks
SET
    end_date = start_date + 7
WHERE
    end_date IS NULL;
```

This query verifies the update:

```
SELECT * FROM tasks;
```

Output:

```
+-----+-----+-----+
| id | title | start_date | end_date |
+-----+-----+-----+
| 1 | Learn MySQL NOT NULL constraint | 2017-02-01 | 2017-02-02 |
| 2 | Check and update NOT NULL constraint to your database | 2017-02-01 | 2017-02-08 |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

Third, add a `NOT NULL` constraint to the `end_date` column using the following [ALTER TABLE](#) statement:

```
ALTER TABLE table_name
CHANGE
    old_column_name
    new_column_name column_definition;
```

In this case, the name of the old and new column names are the same except that the column must have a `NOT NULL` constraint:

```
ALTER TABLE tasks
CHANGE
    end_date
    end_date DATE NOT NULL;
```

Finally, verify the change using the `DESCRIBE` statement:

```
DESC tasks;
```

Output:

```
+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra           |
+-----+-----+-----+-----+
| id         | int        | NO   | PRI  | NULL    | auto_increment |
| title      | varchar(255) | NO   |     | NULL    |                 |
| start_date | date       | NO   |     | NULL    |                 |
| end_date   | date       | NO   |     | NULL    |                 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

The output indicates that the `NOT NULL` constraint was added to the `end_date` column successfully.

Removing a NOT NULL constraint

To drop a `NOT NULL` constraint for a column, you use the `ALTER TABLE..MODIFY` statement:

```
ALTER TABLE table_name
MODIFY column_name column_definition;
```

Note that the column definition (`column_definition`) must restate the original column definition without the `NOT NULL` constraint.

For example, the following statement removes the `NOT NULL` constraint from the `end_date` column in the `tasks` table:

```
ALTER TABLE tasks
MODIFY end_date DATE;
```

Here's the structure of the `tasks` table:

```
DESC tasks;
```

Output:

```
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra           |
+-----+-----+-----+-----+-----+
| id         | int        | NO   | PRI  | NULL    | auto_increment |
| title      | varchar(255) | NO   |      | NULL    |                 |
| start_date | date       | NO   |      | NULL    |                 |
| end_date   | date       | YES  |      | NULL    |                 |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Summary

- Use `NOT NULL` constraint to ensure that a column does not contain any `NULL` values.
- Use `ALTER TABLE ... CHANGE` statement to add a `NOT NULL` constraint to an existing column.
- Use `ALTER TABLE ... MODIFY` to drop a `NOT NULL` constraint from a column.

Was this tutorial helpful?



ADVERTISEMENTS

PREVIOUSLY

[MySQL UNIQUE Constraint](#)

UP NEXT

[MySQL DEFAULT](#)

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

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](#)

SITE LINKS

[Donation](#) ❤

[Contact Us](#)

[About](#)

[Privacy Policy](#)

OTHERS

[MySQL Cheat Sheet](#)

[MySQL Resources](#)

[MySQL Books](#)

[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](#)

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