MySQL UNIQUE Constraint

Summary: in this tutorial, you will learn about MySQL `UNIQUE` constraint and how to use it to enforce the uniqueness of values in a column or a group of columns in a table.

Introduction to MySQL UNIQUE constraint

Sometimes, you want to ensure values in a column or a group of columns are unique. For example, email addresses of users in the `users` table, or phone numbers of customers in the `customers` table should be unique. To enforce this rule, you use a `UNIQUE` constraint.

A `UNIQUE` constraint is an integrity constraint that ensures the uniqueness of values in a column or group of columns. A `UNIQUE` constraint can be either a column constraint or a table constraint.

To define a `UNIQUE` constraint for a column when creating a table, you use the following syntax:

```
CREATE TABLE table_name(
    ...,
    column1 datatype UNIQUE,
    ...
);
```

In this syntax, you include the `UNIQUE` keyword in the definition of the column that you want to enforce the uniqueness.

If you insert or update a value that causes a duplicate in the `column1`, MySQL rejects the change and issues an error.

This `UNIQUE` constraint is a column constraint. And you can use it to enforce the unique rule for one column.

To define a `UNIQUE` constraint for two or more columns, you use the following syntax:

```
CREATE TABLE table_name(
   ...
   column1 datatype,
   column2 datatype,
   ...,
   UNIQUE(column1, column2)
);
```

In this syntax, you add a comma-separated list of columns in parentheses after the `UNIQUE` keyword. In this case, MySQL will use the combination of values in both columns `column1` and `column2` to evaluate the uniqueness.

If you define a `UNIQUE` constraint without specifying a name, MySQL automatically generates a name for it. To define a `UNIQUE` constraint with a name, you use this syntax:

```
[CONSTRAINT constraint_name]
UNIQUE(column_list)
```

In this syntax, you specify the name of the `UNIQUE` constraint after the `CONSTRAINT` keyword.

MySQL UNIQUE constraint example

First, creates a new table named `suppliers` with the two `UNIQUE` constraints:

```
CREATE TABLE suppliers (
    supplier_id INT AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
```

```
    phone VARCHAR(15) NOT NULL UNIQUE,
    address VARCHAR(255) NOT NULL,
    PRIMARY KEY (supplier_id),
    CONSTRAINT uc_name_address UNIQUE (name,address)
);
```

In this example, the first `UNIQUE` constraint is defined for the `phone` column:

```
phone VARCHAR(12) NOT NULL UNIQUE
```

The second `UNIQUE` constraint includes both `name` and `address` columns:

```
CONSTRAINT uc_name_address UNIQUE (name , address)
```

Second, insert a row into the `suppliers` table:

```
INSERT INTO suppliers(name, phone, address)
VALUES( 'ABC Inc',
       '(408)-908-2476',
       '4000 North 1st Street');
```

Third, attempt to insert a different supplier but has the phone number that already exists in the `suppliers` table.

```
INSERT INTO suppliers(name, phone, address)
VALUES( 'XYZ Corporation','(408)-908-2476','3000 North 1st Street');
```

MySQL issued the following error:

```
Error Code: 1062. Duplicate entry '(408)-908-2476' for key 'phone'
```

Fourth, change the phone number to a different one and execute the insert statement again.

```
INSERT INTO suppliers(name, phone, address)
VALUES( 'XYZ Corporation','(408)-908-3333','3000 North 1st Street');
```

Fifth, insert a row into the `suppliers` table with values that already exist in the columns `name` and `address` :

```
INSERT INTO suppliers(name, phone, address)
VALUES( 'ABC Inc',
       '(408)-908-1111',
       '4000 North 1st Street');
```

MySQL issued an error because the `UNIQUE` constraint `uc_name_address` was violated.

```
Error Code: 1062. Duplicate entry 'ABC Inc-4000 North 1st Street' for key 'uc_name_address
```

MySQL UNIQUE constraint & NULL

In MySQL, NULL values are treated as distinct when it comes to unique constraints. Therefore, if you have a column that accepts NULL values, you can insert multiple values into the column.

First, create a new table called `contacts` :

```
CREATE TABLE contacts(
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    phone VARCHAR(20) UNIQUE
)
```

The contacts table has a `phone` column with a UNIQUE constraint. Also, the `phone` column can accept NULL values.

Second, insert some rows into the `contacts` table:

```
INSERT INTO contacts(name, phone)
VALUES
   ('Alice','(408)-102-2456'),
   ('John', NULL),
   ('Jane', NULL);
```

In this example, we can insert two NULL values into the phone column without causing a duplicate.

Third, retrieve data from the `contacts` table:

```
SELECT * FROM contacts;
```

Output:

```
+----+-------+----------------+
| id | name  | phone          |
+----+-------+----------------+
|  1 | Alice | (408)-102-2456 |
|  2 | John  | NULL           |
|  3 | Jane  | NULL           |
+----+-------+----------------+
3 rows in set (0.00 sec)
```

MySQL UNIQUE constraints and indexes

When you define a unique constraint for a column or a group of columns, MySQL creates a corresponding `UNIQUE` index and uses this index to enforce the rule.

The `SHOW CREATE TABLE` statement shows the definition of the `suppliers` table:

```
SHOW CREATE TABLE suppliers;
```

Table	Create Table
suppliers	CREATE TABLE `suppliers` (`supplier_id` int(11) NOT NULL AUTO_INCREMENT, `name` varchar(255) NOT NULL, `phone` varchar(15) NOT NULL, `address` varchar(255) NOT NULL, PRIMARY KEY (`supplier_id`), UNIQUE KEY `phone` (`phone`), UNIQUE KEY `uc_name_address` (`name`,`address`)) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1

The output indicates that MySQL created two `UNIQUE` indexes on the `suppliers` table: `phone` and `uc_name_address`.

The following `SHOW INDEX` statement displays all indexes associated with the `suppliers` table.

```
SHOW INDEX FROM suppliers;
```

Drop a unique constraint

To drop a `UNIQUE` constraint, you can use `DROP INDEX` or `ALTER TABLE` statement:

```
DROP INDEX index_name ON table_name;
```

```
ALTER TABLE table_name
DROP INDEX index_name;
```

For example, the following statement drops the `uc_name_address` constraint on the `suppliers` table:

```
DROP INDEX uc_name_address ON suppliers;
```

Execute the `SHOW INDEX` statement again to verify if the `uc_name_unique` constraint has been removed.

```
SHOW INDEX FROM suppliers;
```

Add new unique constraint

The following `ALTER TABLE ADD CONSTRAINT` adds a unique constraint to a column of an existing table:

```
ALTER TABLE table_name
ADD CONSTRAINT constraint_name
```

```
UNIQUE (column_list);
```

This statement adds a `UNIQUE` constraint `uc_name_address` back to the `suppliers` table:

```
ALTER TABLE suppliers
ADD CONSTRAINT uc_name_address
UNIQUE (name,address);
```

Note that MySQL will not add a unique constraint if the existing data in the columns of specified in the unique constraint does not comply with the uniqueness rule.

Summary

- Use MySQL `UNIQUE` constraint to enforce the uniqueness of values in a column or group of columns of a table.

PREVIOUSLY
MySQL Disable Foreign Key Checks

UP NEXT
MySQL NOT NULL Constraint

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

MANAGING DATABASES

MANAGING 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

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

SITE LINKS

Donation ❤️

Contact Us

About

Privacy Policy

OTHERS

MySQL Cheat Sheet

MySQL Resources

MySQL Books

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