MySQLTUTORIAL

MySQL UNIQUE Constraint

Summary: in this tutorial, you will learn about MySQL `UNIQUE` constraint and how to use `UNIQUE` constraint 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 values in a column or group of columns to be unique. A `UNIQUE` constraint can be either a column constraint or a table constraint.

To define a `UNIQUE` constraint for a column when you create a table (https://www.mysqltutorial.org/mysql-create-table/) , you use this syntax:

```
CREATE TABLE table_name(
    ...,
    column_name data_type UNIQUE,
    ...
);
```

In this syntax, you include the `UNIQUE` keyword in the definition of the column that you want to enforce the uniqueness rule. If you insert (https://www.mysqltutorial.org/mysql-insert-statement.aspx) or update (https://www.mysqltutorial.org/mysql-update-data.aspx) a value that causes duplicate in the `column_name` , 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(
   ...
   column_name1 column_definition,
   column_name2 column_definition,
   ...,
   UNIQUE(column_name1,column_name2)
);
```

In this syntax, you add a comma-separated list of columns in parentheses after the `UNIQUE` keyword. MySQL uses the combination of values in both column `column_name1` and `column_name2` 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 (https://www.mysqltutorial.org/mysql-create-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
```

And the second constraint is for `both` name and `address` columns:

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

Second, insert a row (https://www.mysqltutorial.org/mysql-insert-statement.aspx) 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 an 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 constraints and indexes

When you define a unique constraint, MySQL creates a corresponding `UNIQUE` index (https://www.mysqltutorial.org/mysql-unique/) and uses this index to enforce the rule.

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

```
SHOW CREATE TABLE suppliers;
```

As you can see from the output, MySQL created two `UNIQUE` indexes on the `suppliers` table: `phone` and `uc_name_address`.

The following `SHOW INDEX (https://www.mysqltutorial.org/mysql-index/mysql-show-indexes/)` statement displays all indexes associated with the `suppliers` table.

```
SHOW INDEX FROM suppliers;
```

Drop a unique constraint

To drop a `UNIQUE` constraint, you use can use `DROP INDEX` (https://www.mysqltutorial.org/mysql-index/mysql-drop-index/) or `ALTER TABLE` (https://www.mysqltutorial.org/mysql-alter-table.aspx) 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.

In this tutorial, you have learned how to use the MySQL `UNIQUE` constraint to enforce the uniqueness of values in a column or group of columns of a table.