MySQLTUTORIAL

MySQL Primary Key

Summary: in this tutorial, you will learn how to use **MySQL primary key** constraint to create the primary key for a table.

Introduction to MySQL primary key

A primary key is a column or a set of columns that uniquely identifies each row in the table. The primary key follows these rules:

- A primary key must contain unique values. If the primary key consists of multiple columns, the combination of values in these columns must be unique.
- A primary key column cannot have `NULL` (https://www.mysqltutorial.org/mysql-null/) values. Any attempt to insert (https://www.mysqltutorial.org/mysql-insert-statement.aspx) or update (https://www.mysqltutorial.org/mysql-update-data.aspx) `NULL` to primary key columns will result in an error. Note that MySQL implicitly adds a `NOT NULL` constraint to primary key columns.
- A table can have one an only one primary key.

Because MySQL works faster with integers, the data type (https://www.mysqltutorial.org/mysql-data-types.aspx) of the primary key column should be the integer e.g., `INT` (https://www.mysqltutorial.org/mysql-int/) , `BIGINT` . And you should ensure sure that value ranges of the integer type for the primary key are sufficient for storing all possible rows that the table may have.

A primary key column often has the `AUTO_INCREMENT` (https://www.mysqltutorial.org/mysql-sequence/) attribute that automatically generates a sequential integer whenever you insert a new row (https://www.mysqltutorial.org/mysql-insert-statement.aspx) into the table.

When you define a primary key for a table, MySQL automatically creates an index (https://www.mysqltutorial.org/mysql-index/mysql-create-index/) called `PRIMARY` .

MySQL PRIMARY KEY examples

The `PRIMARY KEY` constraint allows you to define a primary key of a table when you create (https://www.mysqltutorial.org/mysql-create-table/) or alter table.

1) Define a PRIMARY KEY constraint in CREATE TABLE

Typically, you define the primary key for a table in the `CREATE TABLE (https://www.mysqltutorial.org/mysql-create-table/)` statement.

If the primary key has one column, you can use the `PRIMARY KEY` constraint as a column constraint:

```
CREATE TABLE table_name(
    primary_key_column datatype PRIMARY KEY,
    ...
);
```

When the primary key has more than one column, you must use the `PRIMARY KEY` constraint as a table constraint.

```
CREATE TABLE table_name(
    primary_key_column1 datatype,
    primary_key_column2 datatype,
    ...,
    PRIMARY KEY(column_list)
);
```

In this syntax, you separate columns in the `column_list` by commas (,).

The `PRIMARY KEY` table constraint can be used when the primary key has one column:

```
CREATE TABLE table_name (
    primary_key_column datatype,
    ... ,
    PRIMARY KEY(primary_key_column)
);
```

The following example creates a table (https://www.mysqltutorial.org/mysql-create-table/) named `users` whose primary key is the `user_id` column:

```
CREATE TABLE users(
   user_id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(40),
   password VARCHAR(255),
   email VARCHAR(255)
);
```

This statement creates the `roles` table that has the `PRIMARY KEY` constraint as the table constraint:

```
CREATE TABLE roles(
   role_id INT AUTO_INCREMENT,
   role_name VARCHAR(50),
   PRIMARY KEY(role_id)
);
```

In case the primary key consists of multiple columns, you must specify them at the end of the `CREATE TABLE` statement. You put a comma-separated list of primary key columns inside parentheses followed the `PRIMARY KEY` keywords.

The following example creates the `user_roles` table whose primary key consists of two columns: `user_id` and `role_id`. It defines the `PRIMARY KEY` constraint as the table constraint:

```
CREATE TABLE user_roles(
   user_id INT,
   role_id INT,
   PRIMARY KEY(user_id,role_id),
   FOREIGN KEY(user_id)
       REFERENCES users(user_id),
   FOREIGN KEY(role_id)
       REFERENCES roles(role_id)
);
```

Note that the statement also created two foreign key (https://www.mysqltutorial.org/mysql-foreign-key/) constraints.

2) Define PRIMARY KEY constraints using ALTER TABLE

If a table, for some reasons, does not have a primary key, you can use the `ALTER TABLE` (https://www.mysqltutorial.org/mysql-alter-table.aspx) statement to add a primary key to the table as follows:

```
ALTER TABLE table_name
ADD PRIMARY KEY(column_list);
```

The following example adds the `id` column to the primary key.

First, create the `pkdemos` table without a primary key.

```
CREATE TABLE pkdemos(
   id INT,
   title VARCHAR(255) NOT NULL
);
```

Second, add a primary key to the `pkdemos` table using the `ALTER TABLE` statement:

```
ALTER TABLE pkdemos
ADD PRIMARY KEY(id);
```

If you add a primary key to a table that already has data. The data in the column(s), which will be included in the primary key, must be unique and not NULL.

PRIMARY KEY vs. UNIQUE KEY vs. KEY

`KEY` is the synonym for `INDEX`. You use the `KEY` when you want to create an index (https://www.mysqltutorial.org/mysql-index/mysql-create-index/) for a column or a set of columns that is not the part of a primary key or unique key (https://www.mysqltutorial.org/mysql-unique/) .

A `UNIQUE` (https://www.mysqltutorial.org/mysql-unique/) index ensures that values in a column must be unique. Unlike the `PRIMARY` index, MySQL allows `NULL` values in the `UNIQUE` index. In addition, a table

can have multiple `UNIQUE` indexes.

Suppose that `email` and `username` of users in the `users` table must be unique. To enforce thes rules, you can define `UNIQUE` indexes for the `email` and `username` columns as the following statement:

Add a `UNIQUE` index for the `username` column:

```
ALTER TABLE users
ADD UNIQUE INDEX username_unique (username ASC) ;
```

Add a `UNIQUE` index for the `email` column:

```
ALTER TABLE users
ADD UNIQUE INDEX  email_unique (email ASC) ;
```

In this tutorial, you have learned how to create a primary key for a new table or add a primary key to an existing table.