MySQLTUTORIAL

MySQL REPLACE

Summary: in this tutorial, you will learn how to use the MySQL `REPLACE` statement to insert or update data in database tables.

Introduction to MySQL REPLACE statement

The MySQL `REPLACE` statement is an extension to the SQL (http://www.sqltutorial.org) Standard (http://www.sqltutorial.org) . The MySQL `REPLACE` statement works as follows:

Step 1. Insert a new row into the table (https://www.mysqltutorial.org/mysql-insert-statement.aspx) , if a duplicate key error occurs.

Step 2. If the insertion fails due to a duplicate-key error occurs:

- Delete (https://www.mysqltutorial.org/mysql-delete-statement.aspx) the conflicting row that causes the duplicate key error from the table.
- Insert the new row into the table again.

To determine whether the new row that already exists in the table, MySQL uses `PRIMARY KEY (https://www.mysqltutorial.org/mysql-primary-key/)` or `UNIQUE KEY (https://www.mysqltutorial.org/mysql-unique/)` index. If the table does not have one of these indexes, the `REPLACE` works like an `INSERT` statement.

To use the `REPLACE` statement, you need to have at least both `INSERT` and `DELETE` privileges for the table.

> Notice that MySQL has the `REPLACE` string function (https://www.mysqltutorial.org/mysql-string-replace-function.aspx) which is not the `REPLACE` statement covered in this tutorial.

Using MySQL REPLACE to insert a new row

The following illustrates the syntax of the `REPLACE` statement:

```
REPLACE [INTO] table_name(column_list)
VALUES(value_list);
```

It is similar to the INSERT statement except for the keyword REPLACE.

Let's take a look at the following example of using the `REPLACE` statement to see how it works.

First, create a new table (https://www.mysqltutorial.org/mysql-create-table/) named `cities` as follows:

```
CREATE TABLE cities (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    population INT NOT NULL
);
```

Next, insert some rows (https://www.mysqltutorial.org/mysql-insert-multiple-rows/) into the `cities` table:

```
INSERT INTO cities(name,population)
VALUES('New York',8008278),
          ('Los Angeles',3694825),
          ('San Diego',1223405);
```

Then, query data (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) from the `cities` table to verify the insert operation.

```
SELECT * FROM cities;
```

After that, use the `REPLACE` statement to update the population of the `Los Angeles` city to `3696820`.

```
REPLACE INTO cities(id,population)
VALUES(2,3696820);
```

Finally, query the data of the `cities` table again to verify the replacement.

```
SELECT * FROM cities;
```

The value in the `name` column is `NULL` now. The `REPLACE` statement works as follows:

1. First, `REPLACE` statement attempted to insert a new row into `cities` the table. The insertion failed because the id 2 already exists in the `cities` table.
2. Then, `REPLACE` statement deleted the row with id 2 and inserted a new row with the same id 2 and population `3696820`. Because no value is specified for the name column, it was set to `NULL`.

Using MySQL REPLACE statement to update a row

The following illustrates how to use the `REPLACE` statement to update data:

```
REPLACE INTO table
SET column1 = value1,
    column2 = value2;
```

This statement is like the `UPDATE` statement except for the `REPLACE` keyword. In addition, it has no `WHERE (https://www.mysqltutorial.org/mysql-where/)` clause.

This example uses the `REPLACE` statement to update the population of the `Phoenix` city to `1768980`:

```
REPLACE INTO cities
SET id = 4,
    name = 'Phoenix',
    population = 1768980;
```

Unlike the `UPDATE` statement, if you don't specify the value for the column in the `SET` clause, the `REPLACE` statement will use the default value of that column.

```
SELECT * FROM cities;
```

Using MySQL REPLACE to insert data from a SELECT statement

The following illustrates the `REPLACE` statement that inserts data into a table with the data come from a query.

```
REPLACE INTO table_1(column_list)
SELECT column_list
FROM table_2
WHERE where_condition;
```

Note that this form of the `REPLACE` statement is similar to `INSERT INTO SELECT (https://www.mysqltutorial.org/mysql-insert-into-select/)` statement.

The following statement uses the `REPLACE INTO` statement to copy a row within the same table:

```
REPLACE INTO
    cities(name,population)
SELECT
    name,
    population
FROM
   cities
WHERE id = 1;
```

In this tutorial, you've learned different forms of the MySQL `REPLACE` statement to insert or update data in a table.