MySQLTUTORIAL

MySQL UPDATE

Summary: updating data is one of the most important tasks when you work with the database. In this tutorial, you will learn how to use the MySQL `UPDATE` statement to update data in a table.

Introduction to MySQL UPDATE statement

The `UPDATE` statement updates data in a table. It allows you to change the values in one or more columns of a single row or multiple rows.

The following illustrates the basic syntax of the `UPDATE` statement:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_name
SET
    column_name1 = expr1,
    column_name2 = expr2,
    ...
[WHERE
    condition];
```

In this syntax:

- First, specify the name of the table that you want to update data after the `UPDATE` keyword.
- Second, specify which column you want to update and the new value in the `SET` clause. To update values in multiple columns, you use a list of comma-separated assignments by supplying a value in each column's assignment in the form of a literal value, an expression, or a subquery (https://www.mysqltutorial.org/mysql-subquery/) .
- Third, specify which rows to be updated using a condition in the `WHERE (https://www.mysqltutorial.org/mysql-where/)` clause. The `WHERE` clause is optional. If you omit it, the `UPDATE` statement will modify all rows in the table.

Notice that the `WHERE` clause is so important that you should not forget. Sometimes, you may want to update just one row; However, you may forget the `WHERE` clause and accidentally update all rows of the table.

MySQL supports two modifiers in the `UPDATE` statement.

1. The `LOW_PRIORITY` modifier instructs the `UPDATE` statement to delay the update until there is no connection reading data from the table. The `LOW_PRIORITY` takes effect for the storage engines (https://www.mysqltutorial.org/understand-mysql-table-types-innodb-myisam.aspx) that use table-level locking (https://www.mysqltutorial.org/mysql-table-locking/) only such as `MyISAM` , `MERGE` , and `MEMORY` .
2. The `IGNORE` modifier enables the `UPDATE` statement to continue updating rows even if errors occurred. The rows that cause errors such as duplicate-key conflicts are not updated.

MySQL UPDATE examples

Let's practice the `UPDATE` statement.

1) Using MySQL UPDATE to modify values in a single column example

See the following `employees` table from the sample database (https://www.mysqltutorial.org/mysql-sample-database.aspx) .

In this example, we will update the email of `Mary Patterson` to the new email `mary.patterso@classicmodelcars.com` .

First, find Mary's email from the `employees` table using the following `SELECT (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx)` statement:

```
SELECT
    firstname,
    lastname,
    email
FROM
    employees
WHERE
    employeeNumber = 1056;
```

Second, update the email address of `Mary` to the new email `mary.patterson@classicmodelcars.com` :

```
UPDATE employees
SET
    email = 'mary.patterson@classicmodelcars.com'
WHERE
    employeeNumber = 1056;
```

MySQL issued the number of rows affected:

```
1 row(s) affected
```

In this `UPDATE` statement:

- The `WHERE (https://www.mysqltutorial.org/mysql-where/)` clause specifies the row with employee number `1056` will be updated.
- The `SET` clause sets the value of the `email` column to the new email.

Third, execute the `SELECT` statement again to verify the change:

```
SELECT
    firstname,
    lastname,
    email
FROM
    employees
```

```
WHERE
    employeeNumber = 1056;
```

2) Using MySQL UPDATE to modify values in multiple columns

To update values in the multiple columns, you need to specify the assignments in the `SET` clause. For example, the following statement updates both last name and email columns of employee number 1056:

```
UPDATE employees
SET
    lastname = 'Hill',
    email = 'mary.hill@classicmodelcars.com'
WHERE
    employeeNumber = 1056;
```

Let’s verify the changes:

```
SELECT
    firstname,
    lastname,
    email
FROM
    employees
WHERE
    employeeNumber = 1056;
```

3) Using MySQL UPDATE to replace string example

The following example updates the domain parts of emails of all `Sales Reps` with office code `6`:

```
UPDATE employees
SET email = REPLACE(email,'@classicmodelcars.com','@mysqltutorial.org')
WHERE
   jobTitle = 'Sales Rep' AND
   officeCode = 6;
```

In this example, the `REPLACE (https://www.mysqltutorial.org/mysql-string-replace-function.aspx) ()` function replaces `@classicmodelcars.com` in the email column with `@mysqltutorial.org`.

4) Using MySQL UPDATE to update rows returned by a SELECT statement example

You can supply the values for the `SET` clause from a `SELECT` statement that queries data from other tables.

For example, in the `customers` table, some customers do not have any sale representative. The value of the column `saleRepEmployeeNumber` is `NULL` as follows:

```
SELECT
    customername,
    salesRepEmployeeNumber
FROM
    customers
WHERE
    salesRepEmployeeNumber IS NULL;
```

We can take a sale representative and update for those customers.

To do this, we can select a random employee whose job title is `Sales Rep` from the `employees` table and update it for the `employees` table.

This query selects a random (https://www.mysqltutorial.org/select-random-records-database-table.aspx) employee from the table `employees` whose job title is the `Sales Rep`.

```
SELECT
    employeeNumber
FROM
    employees
WHERE
    jobtitle = 'Sales Rep'
ORDER BY RAND()
LIMIT 1;
```

To update the sales representative employee number column in the `customers` table, we place the query above in the `SET` clause of the `UPDATE` statement as follows:

```
UPDATE customers
SET
    salesRepEmployeeNumber = (SELECT
            employeeNumber
        FROM
            employees
        WHERE
            jobtitle = 'Sales Rep'
        ORDER BY RAND()
        LIMIT 1)
WHERE
    salesRepEmployeeNumber IS NULL;
```

If you query data from the `employees` table, you will see that every customer has a sales representative. In other words, the following query returns no row.

```
SELECT
     salesRepEmployeeNumber
FROM
```

```
    customers
WHERE
    salesRepEmployeeNumber IS NULL;
```

In this tutorial, you have learned how to use MySQL `UPDATE` statement to update data in a database table.