MySQLTUTORIAL

MySQL Prepared Statement

Summary: in this tutorial, you will learn how to use **MySQL prepared statement** to make your queries execute faster and more secure.

Introduction to MySQL prepared statement

Prior MySQL version 4.1, a query is sent to the MySQL server in the textual format. In turn, MySQL returns the data to the client using textual protocol. MySQL has to fully parse the query and transforms the result set into a string before returning it to the client.

The textual protocol has serious performance implication. To address this issue, MySQL added a new feature called prepared statement since version 4.1.

The prepared statement takes advantage of client/server binary protocol (https://dev.mysql.com/doc/internals/en/client-server-protocol.html) . It passes the query that contains placeholders (`?`) to the MySQL Server as the following example:

```
SELECT *
FROM products
WHERE productCode = ?;
```

When MySQL executes this query with different `productcode` values, it does not have to fully parse the query. As a result, this helps MySQL execute the query faster, especially when MySQL executes the same query multiple times.

Since the prepared statement uses placeholders (`?`), this helps avoid many variants of SQL injection hence make your application more secure.

MySQL prepared statement usage

In order to use MySQL prepared statement, you use three following statements:

- `PREPARE` – prepare a statement for execution.

- `EXECUTE` – execute a prepared statement prepared by the `PREPARE` statement.
- `DEALLOCATE PREPARE` – release a prepared statement.

The following diagram illustrates how to use a prepared statement:

MySQL prepared statement example

Let's take a look at an example of using the MySQL prepared statement. The following example will use the `products` table from the sample database (https://www.mysqltutorial.org/mysql-sample-database.aspx) .

First, prepare a statement that returns the product code and name of a product specified by product code:

```
PREPARE stmt1 FROM
        'SELECT
            productCode,
            productName
        FROM products
        WHERE productCode = ?';
```

Second, declare a variable named `pc` , stands for product code, and set its value to `'S10_1678'` :

```
SET @pc = 'S10_1678';
```

Third, execute the prepared statement:

```
EXECUTE stmt1 USING @pc;
```

Fourth, assign the pc variable another product code:

```
SET @pc = 'S12_1099';
```

Fifth, execute the prepared statement with the new product code:

```
EXECUTE stmt1 USING @pc;
```

Finally, release the prepared statement:

```
DEALLOCATE PREPARE stmt1;
```

In this tutorial, you have learned how to use the MySQL prepared statement to make your queries execute faster and more secure.