MySQLTUTORIAL

MySQL TRUNCATE TABLE

Summary: in this tutorial, you will learn how to use the MySQL `TRUNCATE TABLE` statement to delete all data in a table.

Introduction to the MySQL TRUNCATE TABLE statement

The MySQL `TRUNCATE TABLE` statement allows you to delete all data in a table.

Logically, the `TRUNCATE TABLE` statement is like a `DELETE (https://www.mysqltutorial.org/mysql-delete-statement.aspx)` statement without a `WHERE (https://www.mysqltutorial.org/mysql-where/)` clause that deletes all rows from a table, or a sequence of `DROP TABLE (https://www.mysqltutorial.org/mysql-drop-table)` and `CREATE TABLE (https://www.mysqltutorial.org/mysql-create-table/)` statements.

However, the `TRUNCATE TABLE` statement is more efficient than the `DELETE` statement because it drops and recreates the table instead of deleting rows one by one.

Here is the basic syntax of the `TRUNCATE TABLE` statement:

```
TRUNCATE [TABLE] table_name;
```

In this syntax, you specify the name of the table which you want to remove all data after the `TRUNCATE TABLE` keywords.

The `TABLE` keyword is optional. However, it is a good practice to use the `TABLE` keyword to distinguish between the `TRUNCATE TABLE` statement and the `TRUNCATE() (https://www.mysqltutorial.org/mysql-math-functions/mysql-truncate/)` function.

If there is any `FOREIGN KEY (https://www.mysqltutorial.org/mysql-foreign-key/)` constraints from other tables which reference the table that you truncate, the `TRUNCATE TABLE` statement will fail.

Because a truncate operation causes an implicit commit, therefore, it cannot be rolled back.

The `TRUNCATE TABLE` statement resets value in the AUTO_INCREMENT column (https://www.mysqltutorial.org/mysql-reset-auto-increment) to its start value if the table has an `AUTO_INCREMENT`

column.

The `TRUNCATE TABLE` statement does not fire `DELETE` triggers associated with the table that is being truncated.

Unlike a `DELETE` statement, the number of rows affected by the `TRUNCATE TABLE` statement is 0, which should be interpreted as no information.

MySQL TRUNCATE TABLE example

Let's take an example of using the `TRUNCATE TABLE` statement.

First, create a new table (https://www.mysqltutorial.org/mysql-create-table/) named `books` for the demonstration:

```
CREATE TABLE books (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL
)  ENGINE=INNODB;
```

Next, insert dummy data to the `books` table by using the following stored procedure (https://www.mysqltutorial.org/mysql-stored-procedure-tutorial.aspx) :

```
DELIMITER $$
CREATE PROCEDURE load_book_data(IN num INT(4))
BEGIN
        DECLARE counter INT(4) DEFAULT 0;
        DECLARE book_title VARCHAR(255) DEFAULT '';

        WHILE counter < num DO
          SET book_title = CONCAT('Book title #',counter);
          SET counter = counter + 1;

          INSERT INTO books(title)
          VALUES(book_title);
        END WHILE;
END$$
```

```
DELIMITER ;
```

Then, load 10,000 rows into the `books` table. It will take a while.

```
CALL load_book_data(10000);
```

After that, check the data in the `books` table:

```
SELECT * FROM books;
```

Finally, use the `TRUNCATE TABLE` statement to delete all rows from the `books` table:

```
TRUNCATE TABLE books;
```

Note that you can compare the performance between the `TRUNCATE TABLE` with the `DELETE` statement.

In this tutorial, you have learned how to use the MySQL `TRUNCATE TABLE` statement to delete all data from a table efficiently, especially for a large table.