MySQLTUTORIAL

MySQL Temporary Table

Summary: in this tutorial, we will discuss **MySQL temporary table** and show you how to create, use and drop temporary tables.

Introduction to MySQL temporary tables

In MySQL, a temporary table is a special type of table that allows you to store a temporary result set, which you can reuse several times in a single session.

A temporary table is very handy when it is impossible or expensive to query data that requires a single `SELECT` (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement with the `JOIN` (https://www.mysqltutorial.org/mysql-join/) clauses. In this case, you can use a temporary table to store the immediate result and use another query to process it.

A MySQL temporary table has the following specialized features:

- A temporary table is created by using `CREATE TEMPORARY TABLE` statement. Notice that the keyword `TEMPORARY` is added between the `CREATE` and `TABLE` keywords.
- MySQL removes the temporary table automatically when the session ends or the connection is terminated. Of course, you can use the `DROP TABLE` (https://www.mysqltutorial.org/mysql-drop-table) statement to remove a temporary table explicitly when you are no longer use it.
- A temporary table is only available and accessible to the client that creates it. Different clients can create temporary tables with the same name without causing errors because only the client that creates the temporary table can see it. However, in the same session, two temporary tables cannot share the same name.
- A temporary table can have the same name as a normal table in a database. For example, if you create a temporary table named `employees` in the sample database (https://www.mysqltutorial.org/mysql-sample-database.aspx) , the existing `employees` table becomes inaccessible. Every query you issue against the `employees` table is now referring to the temporary table `employees`. When you drop the `employees` temporary table, the permanent `employees` table is available and accessible.

> Even though a temporary table can have the same name as a permanent table, it is not recommended. Because this may lead to confusion and potentially cause an unexpected data loss.

For example, in case the connection to the database server is lost and you reconnect to the server automatically, you cannot differentiate between the temporary table and the permanent one. Then, you may issue a `DROP TABLE` statement to remove the permanent table instead of the temporary table, which is not expected. To avoid this issue, you can use the `DROP TEMPORARY TABLE` statement to drop a temporary table.

MySQL CREATE TEMPORARY TABLE statement

The syntax of the `CREATE TEMPORARY TABLE` staetment is similar to the syntax of the `CREATE TABLE` statement except for the `TEMPORARY` keyword:

```
CREATE TEMPORARY TABLE table_name(
   column_1_definition,
   column_2_definition,
   ...,
   table_constraints
);
```

To create a temporary table whose structure is based on an existing table, you cannot use the `CREATE TEMPORARY TABLE ... LIKE` statement. Instead, you use the following syntax:

```
CREATE TEMPORARY TABLE temp_table_name
SELECT * FROM original_table
LIMIT 0;
```

1) Creating a temporary table example

First, create a new temporary table called `credits` that stores customers' credits:

```
CREATE TEMPORARY TABLE credits(
    customerNumber INT PRIMARY KEY,
```

```
    creditLimit DEC(10,2)
);
```

Then, insert rows from the `customers` table into the temporary table `credits`:

```
INSERT INTO credits(customerNumber,creditLimit)
SELECT customerNumber, creditLimit
FROM customers
WHERE creditLimit > 0;
```

2) Creating a temporary table whose structure based on a query example

The following example creates a temporary table that stores the top 10 customers by revenue. The structure of the temporary table is derived from a `SELECT` statement:

```
CREATE TEMPORARY TABLE top_customers
SELECT p.customerNumber,
       c.customerName,
       ROUND(SUM(p.amount),2) sales
FROM payments p
INNER JOIN customers c ON c.customerNumber = p.customerNumber
GROUP BY p.customerNumber
ORDER BY sales DESC
LIMIT 10;
```

Now, you can query data from the `top_customers` temporary table like querying from a permanent table:

```
SELECT
    customerNumber,
    customerName,
    sales
FROM
    top_customers
ORDER BY sales;
```

Dropping a MySQL temporary table

You can use the `DROP TABLE` statement to remove temporary tables however it is good practice to add the `TEMPORARY` keyword as follows:

```
DROP TEMPORARY TABLE table_name;
```

The `DROP TEMPORARY TABLE` statement removes a temporary table only, not a permanent table. It helps you avoid the mistake of dropping a permanent table when you name your temporary table the same as the name of a permanent table

For example, to remove the `topcustomers` temporary table, you use the following statement:

```
DROP TEMPORARY TABLE top_customers;
```

Notice that if you try to remove a permanent table with the `DROP TEMPORARY TABLE` statement, you will get an error message saying that the table that you are trying drop is unknown.

If you develop an application that uses a connection pooling or persistent connections, it is not guaranteed that the temporary tables are removed automatically when your application is terminated. Because the database connection that the application uses may be still open and placed in a connection pool for other clients to reuse later. Therefore, it is a good practice to always remove the temporary tables whenever you are no longer use them.

Checking if a temporary table exists

MySQL does not provide a function or statement to directly check if a temporary table exists. However, we can create a stored procedure that checks if a temporary table exists or not as follows:

```
DELIMITER //
CREATE PROCEDURE check_table_exists(table_name VARCHAR(100))
BEGIN
    DECLARE CONTINUE HANDLER FOR SQLSTATE '42S02' SET @err = 1;
    SET @err = 0;
    SET @table_name = table_name;
    SET @sql_query = CONCAT('SELECT 1 FROM ',@table_name);
    PREPARE stmt1 FROM @sql_query;
    IF (@err = 1) THEN
        SET @table_exists = 0;
    ELSE
        SET @table_exists = 1;
        DEALLOCATE PREPARE stmt1;
    END IF;
END //
DELIMITER ;
```

In this procedure, we try to select data from a temporary table. If the temporary table exists, the `@table_exists` variable is set to 1, otherwise, it sets to 0.

This statement calls the `check_table_exists` to check if the temporary table `credits` exists:

```
CALL check_table_exists('credits');
SELECT @table_exists;
```

Here is the output:

In this tutorial, you have learned about the MySQL temporary tables and how to manage temporary tables such as creating and removing a new temporary table.