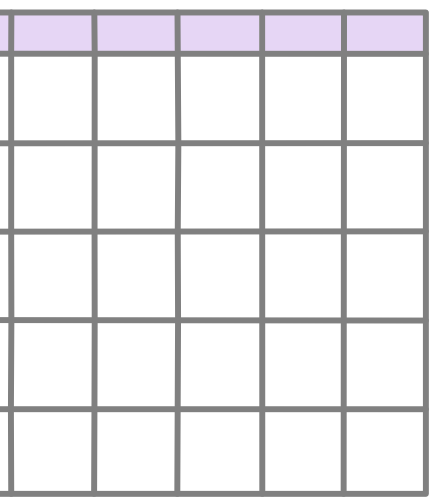
SQL SERIES



How to Create Temporary Table

What is SQL ?

SQL, which stands for Structured Query Language, is a programming language specifically designed for managing and manipulating relational databases.

It provides a standardized way to interact with databases and perform tasks such as querying data, inserting, updating, and deleting records, creating and modifying database structures, and more.

SQL is widely used in the field of data management and plays a crucial role in handling data in various applications and systems.

It allows users to retrieve specific information from databases by using queries, and it provides a powerful set of tools for data manipulation, analysis, and reporting.

What is a Temporary SQL Table?

A temporary SQL table, also known as a temp table, is a table that is created and used within the context of a specific session or transaction in a database management system.

- It is designed to store temporary data that is needed for a short duration and does not require a permanent storage solution.

- Temporary tables are created on-the-fly and are typically used to perform complex calculations, store intermediate results, or manipulate subsets of data during the execution of a query or a series of queries.

- Temporary tables in SQL provide a convenient way to break down complex problems into smaller, more manageable steps.

- They allow for the separation of data processing stages, which can improve performance, enhance code readability, and simplify query logic.

- Temporary tables can be used in various database systems such as MySQL, PostgreSQL, Oracle, SQL Server, and others, although the syntax and features may vary slightly between implementations.

How to Create a Temporary SQL Table

To create a temporary SQL table, we can use the CREATE TABLE statement with the TEMPORARY or TEMP keyword before the table name. Here's an example in SQL:

```
CREATE TEMPORARY TABLE temp_table (
    id INT,
    name VARCHAR(50),
    age INT
);
```

Code explanation:

1. The CREATE TEMPORARY TABLE statement is used to create a temporary table.
2. temp_table is the name given to the temporary table. You can choose any name you like.
3. Inside the parentheses, we define the columns of the temporary table.
4. In this example, the temporary table temp_table has three columns: id of type INT, name of type VARCHAR(50), and age of type INT.
5. We can add more columns as needed, specifying the column name followed by the data type.
6. The temporary table is automatically dropped at the end of the session or when the session is terminated.

SQL Temporary Table Use Cases

Analyzing data subsets using temporary tables

Let's say we have a large dataset and want to perform complex analysis or calculations on a smaller portion of that data.

We can create a temporary table containing only the necessary rows and columns for our analysis. This allows us to focus on a subset of data without modifying the original dataset.

Once our analysis is complete, we can drop the temporary table.

For example:

```
-- Create a temporary table with data subset
CREATE TEMPORARY TABLE subset_data AS
SELECT column1, column2, column3
FROM original_table
WHERE condition;

-- Perform analysis on the subset data
SELECT column1, AVG(column2) AS average_value
FROM subset_data
GROUP BY column1;

-- Drop the temporary table
DROP TABLE subset_data;
```

Improving query performance with temporary tables

You can use temporary tables to optimize complex or resource-intensive queries.

By breaking down a complex query into multiple steps using temporary tables, we can improve query performance by reducing the amount of data processed at each stage or by precomputing intermediate results.

Temporary tables allow us to store and reuse intermediate query results, avoiding redundant computations. Here's an example:

```
-- Create a temporary table to store intermediate results
CREATE TEMPORARY TABLE temp_results AS
SELECT column1, COUNT(*) AS count_value
FROM large_table
WHERE condition1
GROUP BY column1;

-- Use the temporary table to optimize the final query
SELECT column1, column2
FROM temp_results
WHERE count_value > 10
ORDER BY column1;

-- Drop the temporary table
DROP TABLE temp_results;
```

Staging and transforming data using temporary tables

Temporary tables are also useful for staging and transforming data before loading it into permanent tables.

We can create a temporary table, import data from different sources, perform data cleansing, apply transformations, and validate the data before inserting it into the final destination.

Temporary tables provide a flexible and efficient way to process and manipulate data without affecting the original source.

Here's an example:

```
-- Create a temporary table for staging data
CREATE TEMPORARY TABLE staging_table (
    id INT,
    name VARCHAR(50),
    quantity INT);

-- Import and transform data into the staging table
INSERT INTO staging_table (id, name, quantity)
SELECT id, UPPER(name), quantity * 2
FROM external_source;

-- Validate and manipulate data in the staging table
UPDATE staging_table
SET quantity = 0
WHERE quantity < 0;

-- Insert transformed data into the final table
INSERT INTO final_table (id, name, quantity)
SELECT id, name, quantity
FROM staging_table;

-- Drop the temporary table
DROP TABLE staging_table;
```


Differences between Temporary and Permanent Tables in SQL

CRITERIA	TEMPORARY TABLES	PERMANENT TABLES
Lifespan	Exist only for the current session or connection	Persist even after the session or connection is closed.
Data persistence	Data is not persisted beyond the current session	Data is persisted permanently
Storage allocation	Temporary storage is typically allocated in memory or temporary storage space	Permanent storage is allocated on disk or in a database.
Accessibility	Accessible only to the session or connection that created it	Accessible to all users and connections with appropriate privileges.
Naming convention	Temporary table names are often prefixed with a special character or keyword	Permanent table names are not prefixed with any special character or keyword.
Data retention	Data is automatically deleted at the end of the session or connection	Data remains in the table until explicitly deleted or modified.
Indexes and constraints	Temporary tables can have indexes and constraints, but they are typically temporary and are dropped when the table is dropped	Permanent tables can have indexes, constraints, and triggers.

HELPFUL? REPOST

linkedin.com/in/ileonjose