Learn PostgreSQL Recursive Query By Example

Summary: in this tutorial, you will learn about the PostgreSQL recursive query using recursive common table expressions or CTEs.

Introduction to the PostgreSQL recursive query

PostgreSQL provides the `WITH` statement that allows you to construct auxiliary statements for use in a query. These statements are often referred to as common table expressions or CTEs. The CTEs are like temporary tables that only exist during the execution of the query.

A recursive query is a query that refers to a recursive CTE. The recursive queries are useful in many situations such as for querying hierarchical data like organizational structure, bill of materials, etc.

The following illustrates the syntax of a recursive CTE:

```
WITH cte_name(
    CTE_query_definition -- non-recursive term
    UNION [ALL]
    CTE_query definion  -- recursive term
) SELECT * FROM cte_name;
```

A recursive CTE has three elements:

- Non-recursive term: the non-recursive term is a CTE query definition that forms the base result set of the CTE structure.
- Recursive term: the recursive term is one or more CTE query definitions joined with the non-recursive term using the UNION or UNION ALL operator. The recursive term references to the CTE name itself.
- Termination check: the recursion stops when no rows are returned from the previous iteration.

PostgreSQL executes a recursive CTE in the following sequence:

1. Execute non-recursive term to create the base result set (R0).
2. Execute recursive term with Ri as an input to return the result set Ri+1 as the output.
3. Repeat step 2 until an empty set is returned. (termination check)
4. Return the final result set that is a UNION or UNION ALL of the result set R0, R1, … Rn

PostgresQL recursive query example

We will create a new table to demonstrate the PostgreSQL recursive query.

```
CREATE TABLE employees (
 employee_id serial PRIMARY KEY,
 full_name VARCHAR NOT NULL,
 manager_id INT
);
```

The `employees` table consists of three columns: employee_id, manager_id, and full name. The manager_id column specifies the manager id of an employee.

The following statement inserts sample data into the `employees` table.

```
Click To Expand Code
```

The following query returns all subordinates of the manager with the id 2.

```
WITH RECURSIVE subordinates AS (
 SELECT
 employee_id,
 manager_id,
 full_name
 FROM
 employees
 WHERE
 employee_id = 2
 UNION
 SELECT
 e.employee_id,
 e.manager_id,
 e.full_name
 FROM
 employees e
 INNER JOIN subordinates s ON s.employee_id = e.manager_id
) SELECT
 *
FROM
 subordinates;
```

How it works.

- The recursive CTE, subordinates, defines one non-recursive term and one recursive term.
- The non-recursive term returns the base result set R0 that is the employee with the id 2.

```
 employee_id | manager_id |  full_name
-------------+------------+-------------
           2 |          1 | Megan Berry
```

The recursive term returns the direct subordinate(s) of the employee id 2. This is the result of joining between the employees table and the subordinates CTE. The first iteration of the recursive term returns the following result set:

```
 employee_id | manager_id |    full_name
-------------+------------+-----------------
           6 |          2 | Bella Tucker
           7 |          2 | Ryan Metcalfe
           8 |          2 | Max Mills
           9 |          2 | Benjamin Glover
```

PostgreSQL executes the recursive term repeatedly. The second iteration of the recursive member uses the result set above step as the input value, and returns this result set:

```
 employee_id | manager_id |    full_name
-------------+------------+-----------------
          16 |          7 | Piers Paige
          17 |          7 | Ryan Henderson
          18 |          8 | Frank Tucker
          19 |          8 | Nathan Ferguson
          20 |          8 | Kevin Rampling
```

The third iteration returns an empty result set because there is no employee reporting to the employee with the id 16, 17, 18, 19 and 20.

PostgreSQL returns the final result set that is the union of all result sets in the first and second iterations generated by the non-recursive and recursive terms.

```
 employee_id | manager_id |    full_name
-------------+------------+-----------------
           2 |          1 | Megan Berry
           6 |          2 | Bella Tucker
           7 |          2 | Ryan Metcalfe
           8 |          2 | Max Mills
           9 |          2 | Benjamin Glover
          16 |          7 | Piers Paige
          17 |          7 | Ryan Henderson
          18 |          8 | Frank Tucker
          19 |          8 | Nathan Ferguson
          20 |          8 | Kevin Rampling
(10 rows)
```

In this tutorial, you have learned how to use the recursive CTEs to construct the PostgreSQL recursive queries.

« Previous Tutorial:
How to Generate a Random Number in A Range

Next Tutorial:
How To Delete Duplicate Rows in PostgreSQL »

Search this website ...

POSTGRESQL QUICK START

POSTGRESQL FUNDAMENTALS

MANAGING TABLE STRUCTURE

PostgreSQL Create Table

PostgreSQL Alter Table

PostgreSQL Drop Table

PostgreSQL Truncate Table

PostgreSQL CHECK Constraint

PostgreSQL Not-Null Constraint

PostgreSQL Foreign Key

PostgreSQL Primary Key

PostgreSQL UNIQUE Constraint

POSTGRESQL VIEWS

Managing PostgreSQL Views

Creating Updatable Views

PostgreSQL Materialized Views

The WITH CHECK OPTION Views

PostgreSQL Recursive View

POSTGRESQL TRIGGERS

Introduction to Trigger

Creating A Trigger

Managing PostgreSQL Triggers

ABOUT POSTGRESQL TUTORIAL

PostgreSQLTutorial.com is a website dedicated to developers and database administrators who are working on PostgreSQL database management system.

We constantly publish useful PostgreSQL tutorials to keep you up-to-date with the latest PostgreSQL features and technologies. All PostgreSQL tutorials are simple, easy-to-follow and practical.

RECENT POSTGRESQL TUTORIALS

PostgreSQL ANY Operator

PostgreSQL EXISTS

How To Delete Duplicate Rows in PostgreSQL

PostgreSQL TO_CHAR Function

PostgreSQL TO_NUMBER Function

PostgreSQL TO_TIMESTAMP Function

PostgreSQL CEIL Function

PostgreSQL MOD Function

PostgreSQL FLOOR Function

PostgreSQL ABS Function

MORE TUTORIALS

PostgreSQL Cheat Sheet

PostgreSQL Administration

PostgreSQL PHP

PostgreSQL Python

PostgreSQL JDBC

PostgreSQL Resources

SITE INFO

Home

About Us

Contact Us

Privacy Policy