SQLSERVER TUTORIAL.NET

SQL Server Recursive CTE

Summary: in this tutorial, you will learn how to use the SQL Server recursive CTE to query hierarchical data.

Introduction to SQL Server recursive CTE

A recursive common table expression (https://www.sqlservertutorial.net/sql-server-basics/sql-server-cte/) (CTE) is a CTE that references itself. By doing so, the CTE repeatedly executes, returns subsets of data, until it returns the complete result set.

A recursive CTE is useful in querying hierarchical data such as organization charts where one employee reports to a manager or multi-level bill of materials when a product consists of many components, and each component itself also consists of many other components.

The following shows the syntax of a recursive CTE:

```
WITH expression_name (column_list)
AS
(
    -- Anchor member
    initial_query
    UNION ALL
    -- Recursive member that references expression_name.
    recursive_query
)
-- references expression name
SELECT *
FROM   expression_name
```

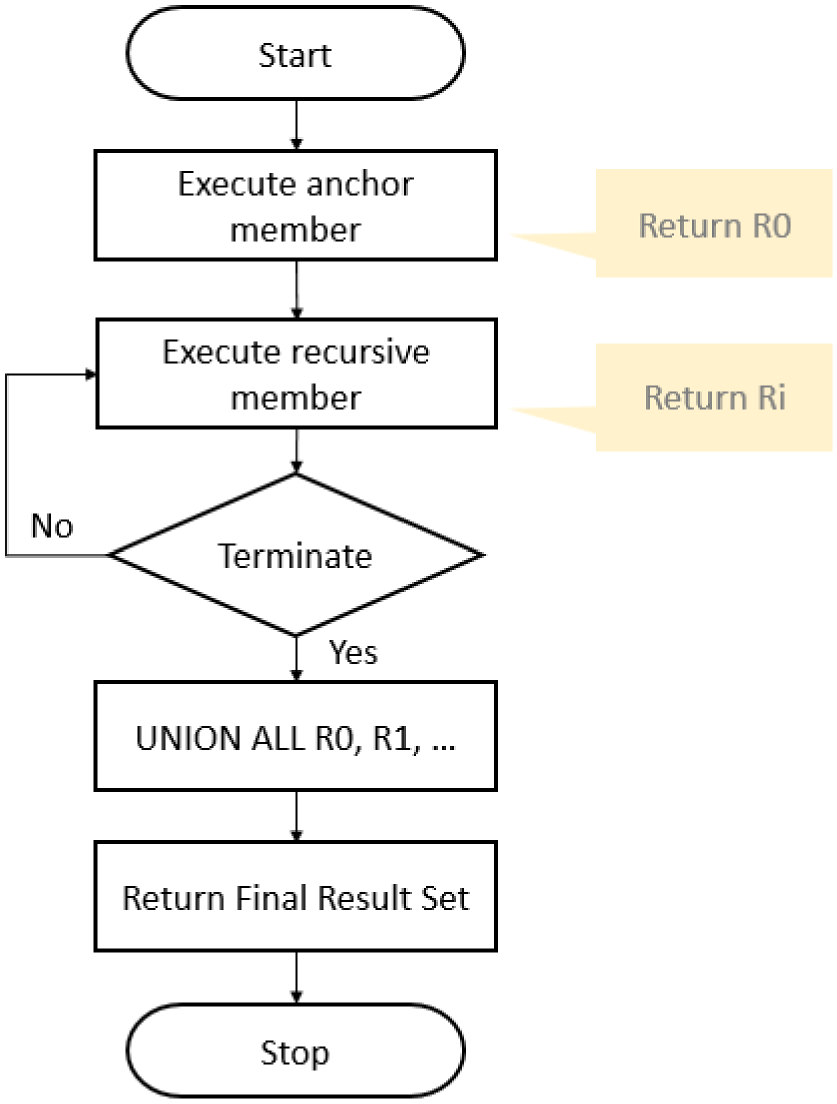
In general, a recursive CTE has three parts:

1. An initial query that returns the base result set of the CTE. The initial query is called an anchor member.
2. A recursive query that references the common table expression, therefore, it is called the recursive member. The recursive member is union-ed with the anchor member using the `UNION ALL` (https://www.sqlservertutorial.net/sql-server-basics/sql-server-union/) operator.
3. A termination condition specified in the recursive member that terminates the execution of the recursive member.

The execution order of a recursive CTE is as follows:

> First, execute the anchor member to form the base result set (R0), use this result for the next iteration.
>
> Second, execute the recursive member with the input result set from the previous iteration (Ri-1) and return a sub-result set (Ri) until the termination condition is met.
>
> Third, combine all result sets R0, R1, … Rn using `UNION ALL` (https://www.sqlservertutorial.net/sql-server-basics/sql-server-union/) operator to produce the final result set.

The following flowchart illustrates the execution of a recursive CTE:



SQL Server Recursive CTE examples

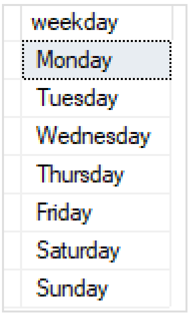
Let's take some examples of using recursive CTEs

A) Simple SQL Server recursive CTE example

This example uses a recursive CTE to returns weekdays from `Monday` to `Saturday` :

```
WITH cte_numbers(n, weekday)
AS (
    SELECT
        0,
        DATENAME(DW, 0)
    UNION ALL
    SELECT
        n + 1,
        DATENAME(DW, n + 1)
    FROM
        cte_numbers
    WHERE n < 6
)
SELECT
    weekday
FROM
    cte_numbers;
```

Here is the result set:



```
weekday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
```

In this example:

The [DATENAME()](https://www.sqlservertutorial.net/sql-server-date-functions/sql-server-datename-function/) (https://www.sqlservertutorial.net/sql-server-date-functions/sql-server-datename-function/) function returns the name of the weekday based on a weekday number.

The anchor member returns the `Monday`

```
SELECT
    0,
    DATENAME(DW, 0)
```

The recursive member returns the next day starting from the `Tuesday` till `Sunday`.

```
    SELECT
        n + 1,
        DATENAME(DW, n + 1)
    FROM
```

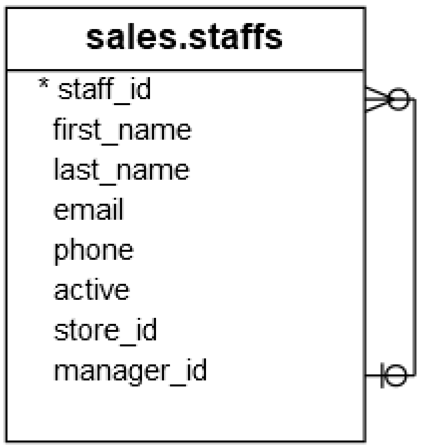
```
        cte_numbers
    WHERE n < 6
```

The condition in the WHERE (https://www.sqlservertutorial.net/sql-server-basics/sql-server-where/) clause is the termination condition that stops the execution of the recursive member when n is 6

```
n < 6
```

B) Using a SQL Server recursive CTE to query hierarchical data

See the following `sales.staffs` table from the sample database (https://www.sqlservertutorial.net/sql-server-sample-database/) :

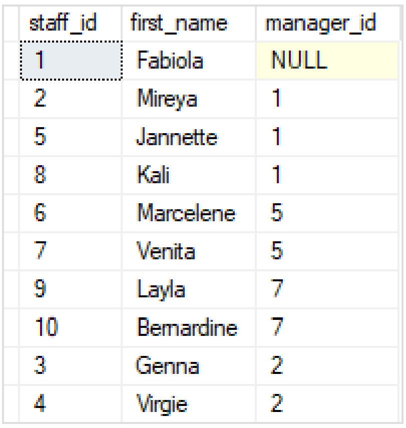


In this table, a staff reports to zero or one manager. A manager may have zero or more staffs. The top manager has no manager. The relationship is specified in the values of the `manager_id` column. If a staff does not report to any staff (in case of the top manager), the value in the `manager_id` is NULL.

This example uses a recursive CTE to get all subordinates of the top manager who does not have a manager (or the value in the `manager_id` column is NULL):

```
WITH cte_org AS (
    SELECT
        staff_id,
        first_name,
        manager_id
    FROM
        sales.staffs
    WHERE manager_id IS NULL
    UNION ALL
    SELECT
        e.staff_id,
        e.first_name,
        e.manager_id
    FROM
        sales.staffs e
        INNER JOIN cte_org o
            ON o.staff_id = e.manager_id
)
SELECT * FROM cte_org;
```

Here is the output:



staff_id	first_name	manager_id
1	Fabiola	NULL
2	Mireya	1
5	Jannette	1
8	Kali	1
6	Marcelene	5
7	Venita	5
9	Layla	7
10	Bernardine	7
3	Genna	2
4	Virgie	2

In this example, the anchor member gets the top manager and the recursive query returns subordinates of the top managers and subordinates of the top manager, and so on.

In this tutorial, you have learned how to use the SQL Server recursive CTE to query hierarchical data.