MySQLTUTORIAL

A Definitive Guide To MySQL Recursive CTE

Summary: in this tutorial, you will learn about MySQL recursive CTE and how to use it to traverse hierarchical data.

Notice that common table expression (https://www.mysqltutorial.org/mysql-cte/) or CTE only available in MySQL version 8.0 or later. Therefore, you should have the right version of MySQL installed in order to use the statements in this tutorial.

Introduction to MySQL recursive CTE

A recursive common table expression (https://www.mysqltutorial.org/mysql-cte/) (CTE) is a CTE that has a subquery which refers to the CTE name itself. The following illustrates the syntax of a recursive CTE

```
WITH RECURSIVE cte_name AS (
    initial_query  -- anchor member
    UNION ALL
    recursive_query -- recursive member that references to the CTE name
)
SELECT * FROM cte_name;
```

A recursive CTE consists of three main parts:

- An initial query (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) that forms the base result set of the CTE structure. The initial query part is referred to as an anchor member.
- A recursive query part is a query that references to the CTE name, therefore, it is called a recursive member. The recursive member is joined with the anchor member by a `UNION ALL` `(https://www.mysqltutorial.org/sql-union-mysql.aspx)` or `UNION DISTINCT` operator.
- A termination condition that ensures the recursion stops when the recursive member returns no row.

The execution order of a recursive CTE is as follows:

1. First, separate the members into two: anchor and recursive members.
2. Next, execute the anchor member to form the base result set (`R0`) and use this base result set for the next iteration.
3. Then, execute the recursive member with `Ri` result set as an input and make `Ri+1` as an output.
4. After that, repeat the third step until the recursive member returns an empty result set, in other words, the termination condition is met.
5. Finally, combine result sets from R0 to Rn using `UNION ALL` operator.

Recursive member restrictions

The recursive member must not contain the following constructs:

- Aggregate functions e.g., MAX (https://www.mysqltutorial.org/mysql-max-function/) , MIN (https://www.mysqltutorial.org/mysql-min/) , SUM (https://www.mysqltutorial.org/mysql-sum/) , AVG (https://www.mysqltutorial.org/mysql-avg/) , COUNT (https://www.mysqltutorial.org/mysql-count/) , etc.
- GROUP BY (https://www.mysqltutorial.org/mysql-group-by.aspx) clause
- ORDER BY (https://www.mysqltutorial.org/mysql-order-by/) clause
- LIMIT (https://www.mysqltutorial.org/mysql-limit.aspx) clause
- DISTINCT (https://www.mysqltutorial.org/mysql-distinct.aspx)

Note that the above constraint does not apply to the anchor member. Also, the prohibition on `DISTINCT` applies only when you use `UNION` operator. In case you use the `UNION DISTINCT` operator, the `DISTINCT` is permitted.

In addition, the recursive member can only reference the CTE name once and in its `FROM` clause, not in any subquery (https://www.mysqltutorial.org/mysql-subquery/) .

Simple MySQL recursive CTE example

See the following simple recursive CTE example:

```
WITH RECURSIVE cte_count (n)
AS (
        SELECT 1
        UNION ALL
        SELECT n + 1
        FROM cte_count
        WHERE n < 3
    )
SELECT n
FROM cte_count;
```

In this example, the following query:

```
SELECT 1
```

is the anchor member that returns 1 as the base result set.

The following query

```
SELECT n + 1
FROM cte_count
WHERE n < 3
```

is the recursive member because it references to the name of the CTE which is `cte_count`.

The expression `n < 3` in the recursive member is the termination condition. Once n equals 3, the recursive member returns an empty set that will stop the recursion.

The following picture illustrates the elements of the CTE above:

The recursive CTE returns the following output:

The execution steps of the recursive CTE is as follows:

1. First, separate the anchor and recursive members.
2. Next, the anchor member forms the initial row (`SELECT 1`) therefore the first iteration produces 1 + 1 = 2 with n = 1.
3. Then, the second iteration operates on the output of the first iteration (2) and produces 2 + 1 = 3 with n = 2.
4. After that, before the third operation (n = 3), the termination condition (`n < 3`) is met therefore the query stops.
5. Finally, combine all result sets 1, 2 and 3 using the `UNION ALL` operator

Using MySQL recursive CTE to traverse the hierarchical data

We will use the `employees` table in the classicmodels sample database (https://www.mysqltutorial.org/mysql-sample-database.aspx) for the demonstration.

The `employees` table has the `reportsTo` column that references to the `employeeNumber` column. The `reportsTo` column stores the ids of managers. The top manager does not report to anyone in the company's organization structure, therefore, the value in the `reportsTo` column is `NULL (https://www.mysqltutorial.org/mysql-null/)` .

You can apply the recursive CTE to query the whole organization structure in the top-down manner as follows:

```
WITH RECURSIVE employee_paths AS
  ( SELECT employeeNumber,
           reportsTo managerNumber,
           officeCode,
           1 lvl
   FROM employees
   WHERE reportsTo IS NULL
     UNION ALL
     SELECT e.employeeNumber,
            e.reportsTo,
            e.officeCode,
            lvl+1
     FROM employees e
     INNER JOIN employee_paths ep ON ep.employeeNumber = e.reportsTo )
SELECT employeeNumber,
       managerNumber,
       lvl,
       city
FROM employee_paths ep
```

```
INNER JOIN offices o USING (officeCode)
ORDER BY lvl, city;
```

Let’s break the query into smaller parts to make it easier to understand.

First, form the anchor member by using the following query:

```
SELECT
    employeeNumber,
    reportsTo managerNumber,
    officeCode
FROM
    employees
WHERE
    reportsTo IS NULL
```

This query (anchor member) returns the top manager whose `reportsTo` is `NULL`.

Second, make the recursive member by reference to the CTE name, which is `employee_paths` in this case:

```
SELECT
    e.employeeNumber,
    e.reportsTo,
    e.officeCode
FROM
    employees e
INNER JOIN employee_paths ep
    ON ep.employeeNumber = e.reportsTo
```

This query (recursive member) returns all direct reports of the manager(s) until there are no more direct reports. The if the recursive member returns no direct reports, the recursion stops.

Third, the query that uses the `employee_paths` CTE joins the result set returned by the CTE with the `offices` table to make the final result set.

The following is the output of the query:

In this tutorial, you have learned about MySQL recursive CTE and how to use it to traverse hierarchical data.