https://dwgeek.com/redshift-recursive-query-alternative-example.html/

**Redshift Recursive Query Example**

* Post author:[Vithal S](https://dwgeek.com/author/vithal/)
* Post last modified:January 20, 2022
* Post category:[Redshift](https://dwgeek.com/category/redshift/)
* Reading time:6 mins read

You can use recursive query to query hierarchies of data, such as an organizational structure, bill-of-materials, and document hierarchy. Redshift does [not support all features](https://docs.aws.amazon.com/redshift/latest/dg/c_unsupported-postgresql-features.html) that are supported in PostgreSQL. One of such features is Recursive CTE or VIEWS. Redshift does not support either [WITH RECURSIVE Clause](https://dwgeek.com/amazon-redshift-with-clause-syntax-usage-examples.html/) or using the RECURSIVE Clause in a CREATE VIEW Statement. In this article, we will check **Redshift Recursive Query Alternative with an working example**.

**Redshift Recursive Query**

Amazon Redshift, a fully-managed cloud data warehouse, now adds support for Recursive Common Table Expression (CTE) to analyze hierarchical data, such as organizational charts where employees reports to other employees (managers), or multi-level product orders where a product consists of many components, which in turn consist of other components.

Following is the syntax of recursive CTE in Redshift.

[ WITH [RECURSIVE] common\_table\_expression [, common\_table\_expression , ...] ] SELECT statement;

For example, you can use the following recursive query for hierarchical data.

with recursive org\_cte(id, name, manager\_id, level) as

( select id, name, manager\_id, 1 as level

from employee

where name = 'John'

union all

select e.id, e.name, e.manager\_id, level + 1

from employee e, john\_org j

where e.manager\_id = j.id and level < 4

)

select id, name, manager\_id from org\_cte order by manager\_id;

**Redshift Recursive CTE Alternative**

In the other RDBMS such as [Teradata](https://dwgeek.com/teradata-with-clause-syntax-usage-and-examples.html/) or [Snowflake](https://dwgeek.com/snowflake-with-clause-syntax-usage-and-examples.html/) you can specify a recursive query by preceding a query with the **WITH RECURSIVE clause** or create a **CREATE VIEW statement.**

Amazon Redshift does not support recursive CTEs, you have to use [Redshift union all set operators](https://dwgeek.com/redshift-set-operators-union-except-minus-intersect.html/) or [inner join](https://dwgeek.com/different-redshift-join-types-examples.html/) approach if you know the depth of the recursive query hierarchy.

But, if you don’t know the depth of the recursive query hierarchy, you have to use [Redshift stored procedure](https://dwgeek.com/working-with-amazon-redshift-stored-procedure.html/) to identify the depth dynamically.

**Redshift Recursive Query Alternative using Stored Procedure**

To implement the Redshift recursive CTE, we have already created emp table. We need to identify the employee and manager hierarchy. Here is the content of table.

|  |  |
| --- | --- |
| **EMPLOYEE\_NUMBER** | **MANAGER\_EMPLOYEE\_NUMBER** |
| 801 | NULL |
| 1016 | 801 |
| 1003 | 801 |
| 1019 | 801 |
| 1010 | 1003 |
| 1004 | 1003 |
| 1001 | 1003 |
| 1012 | 1004 |
| 1002 | 1004 |
| 1015 | 1004 |
| 1008 | 1019 |
| 1006 | 1019 |
| 1014 | 1019 |
| 1011 | 1019 |

Following is the example of Redshift recursive CTE using [stored procedure](https://dwgeek.com/working-with-amazon-redshift-stored-procedure.html/).

create or replace procedure Rec\_test2( result\_set INOUT REFCURSOR)

AS $$

DECLARE

df\_cnt int := 1;

cnt int :=1;

a int;

tblnm varchar(50) := '';

tblnm1 varchar(50) := '';

qry varchar(2500) := '';

BEGIN

-- Create initial temp table

create temp table temp\_tab0 as (SELECT employee\_number from employee\_rec\_test where manager\_employee\_number=801);

while (df\_cnt != 0 )

loop

tblnm := 'temp\_tab'||(cnt-1);

tblnm1 := 'temp\_tab'||cnt;

execute 'select \* from( select indirect.employee\_number

FROM '||quote\_ident(tblnm)||' direct, employee\_rec\_test indirect

WHERE direct.employee\_number = indirect.manager\_employee\_number) a';

GET DIAGNOSTICS df\_cnt := ROW\_COUNT;

if (df\_cnt!=0)

then

execute 'create temp table '||quote\_ident(tblnm1)||' as select indirect.employee\_number

FROM '||quote\_ident(tblnm)||' direct, employee\_rec\_test indirect

WHERE direct.employee\_number = indirect.manager\_employee\_number';

end if;

cnt := cnt + 1;

end loop;

for a in 0..(cnt-2)

loop

if (a=0)

then

qry := qry || 'select employee\_number from temp\_tab' || a;

else

qry := qry || ' union select employee\_number from temp\_tab' || a;

end if;

end loop;

-- Return Result set

--OPEN result\_set for

execute 'create table result\_rec as '|| qry;

END;

$$ LANGUAGE plpgsql;

Following is the result.

call Rec\_test2('newcursor');

select \* from result\_rec order by 1;

employee\_number

1001

1002

1003

1004

1006

1008

1010

1011

1012

1014

1015

1016

1019

Example, is borrowed from Teradata Recursive CTE example. [Teradata Recursive Queries](https://docs.teradata.com/reader/m~O~fVLqvU~MIZ5ZcaXIhg/d86vaCWs7C1XZdknCedj_A).

Related Articles,

* [QUALIFY Clause in Redshift – Alternative and Examples](https://dwgeek.com/qualify-clause-in-redshift-alternative-and-examples.html/)

Hope this helps