A common table expression (CTE) provides the significant advantage of being able to reference itself, thereby creating a recursive CTE. A recursive CTE is one in which an initial CTE is repeatedly executed to return subsets of data until the complete result set is obtained.

A query is referred to as a recursive query when it references a recursive CTE. Returning hierarchical data is a common use of recursive queries, for example: Displaying employees in an organizational chart, or data in a bill of materials scenario in which a parent product has one or more components and those components may, in turn, have subcomponents or may be components of other parents.

A recursive CTE can greatly simplify the code required to run a recursive query within a SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. In earlier versions of SQL Server, a recursive query usually requires using temporary tables, cursors, and logic to control the flow of the recursive steps. For more information about common table expressions, see Using Common Table Expressions.

**ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/collapse.gifStructure of a Recursive CTE**

The structure of the recursive CTE in Transact-SQL is similar to recursive routines in other programming languages. Although a recursive routine in other languages returns a scalar value, a recursive CTE can return multiple rows.

A recursive CTE consists of three elements:

1. Invocation of the routine.  
     
   The first invocation of the recursive CTE consists of one or more *CTE\_query\_definitions* joined by UNION ALL, UNION, EXCEPT, or INTERSECT operators. Because these query definitions form the base result set of the CTE structure, they are referred to as anchor members.  
     
   *CTE\_query\_definitions* are considered anchor members unless they reference the CTE itself. All anchor-member query definitions must be positioned before the first recursive member definition, and a UNION ALL operator must be used to join the last anchor member with the first recursive member.
2. Recursive invocation of the routine.  
     
   The recursive invocation includes one or more *CTE\_query\_definitions* joined by UNION ALL operators that reference the CTE itself. These query definitions are referred to as recursive members.
3. Termination check.   
     
   The termination check is implicit; recursion stops when no rows are returned from the previous invocation.

|  |
| --- |
| **ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/note.gifNote:** |
| An incorrectly composed recursive CTE may cause an infinite loop. For example, if the recursive member query definition returns the same values for both the parent and child columns, an infinite loop is created. When testing the results of a recursive query, you can limit the number of recursion levels allowed for a specific statement by using the MAXRECURSION hint and a value between 0 and 32,767 in the OPTION clause of the INSERT, UPDATE, DELETE, or SELECT statement. For more information, see Query Hints (Transact-SQL) and WITH common\_table\_expression (Transact-SQL). |

**Pseudocode and Semantics**

The recursive CTE structure must contain at least one anchor member and one recursive member. The following pseudocode shows the components of a simple recursive CTE that contains a single anchor member and single recursive member.

WITH cte\_name ( column\_name [,...n] )

AS

(

CTE\_query\_definition –- Anchor member is defined.

UNION ALL

CTE\_query\_definition –- Recursive member is defined referencing cte\_name.

)

-- Statement using the CTE

SELECT \*

FROM cte\_name

The semantics of the recursive execution is as follows:

1. Split the CTE expression into anchor and recursive members.
2. Run the anchor member(s) creating the first invocation or base result set (T0).
3. Run the recursive member(s) with Ti as an input and Ti+1 as an output.
4. Repeat step 3 until an empty set is returned.
5. Return the result set. This is a UNION ALL of T0 to Tn.

**ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/collapse.gifExample**

The following example shows the semantics of the recursive CTE structure by returning a hierarchical list of employees, starting with the highest ranking employee, in the Adventure Works Cycles company. A walkthrough of the code execution follows the example.

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/copycode.gifCopy Code |
| -- Create an Employee table.  CREATE TABLE dbo.MyEmployees  (  EmployeeID smallint NOT NULL,  FirstName nvarchar(30) NOT NULL,  LastName nvarchar(40) NOT NULL,  Title nvarchar(50) NOT NULL,  DeptID smallint NOT NULL,  ManagerID int NULL,  CONSTRAINT PK\_EmployeeID PRIMARY KEY CLUSTERED (EmployeeID ASC)  );  -- Populate the table with values.  INSERT INTO dbo.MyEmployees VALUES  (1, N'Ken', N'Snchez', N'Chief Executive Officer',16,NULL)  ,(273, N'Brian', N'Welcker', N'Vice President of Sales',3,1)  ,(274, N'Stephen', N'Jiang', N'North American Sales Manager',3,273)  ,(275, N'Michael', N'Blythe', N'Sales Representative',3,274)  ,(276, N'Linda', N'Mitchell', N'Sales Representative',3,274)  ,(285, N'Syed', N'Abbas', N'Pacific Sales Manager',3,273)  ,(286, N'Lynn', N'Tsoflias', N'Sales Representative',3,285)  ,(16, N'David',N'Bradley', N'Marketing Manager', 4, 273)  ,(23, N'Mary', N'Gibson', N'Marketing Specialist', 4, 16); | |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/copycode.gifCopy Code |
| USE AdventureWorks2008R2;  GO  WITH DirectReports (ManagerID, EmployeeID, Title, DeptID, Level)  AS  (  -- Anchor member definition  SELECT e.ManagerID, e.EmployeeID, e.Title, edh.DepartmentID,  0 AS Level  FROM dbo.MyEmployees AS e  INNER JOIN HumanResources.EmployeeDepartmentHistory AS edh  ON e.EmployeeID = edh.BusinessEntityID AND edh.EndDate IS NULL  WHERE ManagerID IS NULL  UNION ALL  -- Recursive member definition  SELECT e.ManagerID, e.EmployeeID, e.Title, edh.DepartmentID,  Level + 1  FROM dbo.MyEmployees AS e  INNER JOIN HumanResources.EmployeeDepartmentHistory AS edh  ON e.EmployeeID = edh.BusinessEntityID AND edh.EndDate IS NULL  INNER JOIN DirectReports AS d  ON e.ManagerID = d.EmployeeID  )  -- Statement that executes the CTE  SELECT ManagerID, EmployeeID, Title, DeptID, Level  FROM DirectReports  INNER JOIN HumanResources.Department AS dp  ON DirectReports.DeptID = dp.DepartmentID  WHERE dp.GroupName = N'Sales and Marketing' OR Level = 0;  GO | |

**Example Code Walkthrough**

1. The recursive CTE, DirectReports, defines one anchor member and one recursive member.
2. The anchor member returns the base result set T0. This is the highest ranking employee in the company; that is, an employee who does not report to a manager.   
     
   Here is the result set returned by the anchor member:

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/copycode.gifCopy Code |
| ManagerID EmployeeID Title                         Level  --------- ---------- ----------------------------- ------  NULL      1          Chief Executive Officer        0 | |

1. The recursive member returns the direct subordinate(s) of the employee in the anchor member result set. This is achieved by a join operation between the Employee table and the DirectReports CTE. It is this reference to the CTE itself that establishes the recursive invocation. Based on the employee in the CTE DirectReports as input (Ti), the join (MyEmployees.ManagerID = DirectReports.EmployeeID) returns as output (Ti+1), the employees who have (Ti) as their manager. Therefore, the first iteration of the recursive member returns this result set:

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/copycode.gifCopy Code |
| ManagerID EmployeeID Title                         Level  --------- ---------- ----------------------------- ------  1         273        Vice President of Sales       1 | |

1. The recursive member is activated repeatedly. The second iteration of the recursive member uses the single-row result set in step 3 (containing EmployeeID 273) as the input value, and returns this result set:

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/copycode.gifCopy Code |
| ManagerID EmployeeID Title                         Level  --------- ---------- ----------------------------- ------  273       16         Marketing Manager             2  273       274        North American Sales Manager  2  273       285        Pacific Sales Manager         2 | |

1. The third iteration of the recursive member uses the result set above as the input value, and returns this result set:

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/copycode.gifCopy Code |
| ManagerID EmployeeID Title                         Level  --------- ---------- ----------------------------- ------  16        23         Marketing Specialist          3  274       275        Sales Representative          3  274       276        Sales Representative          3  285       286        Sales Representative          3 | |

1. The final result set returned by the running query is the union of all result sets generated by the anchor and recursive members.   
     
   Here is the complete result set returned by the example:

|  |  |
| --- | --- |
|  | ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de_1devconc/local/copycode.gifCopy Code |
| ManagerID EmployeeID Title                         Level  --------- ---------- ----------------------------- ------  NULL      1          Chief Executive Officer       0  1         273        Vice President of Sales       1  273       16         Marketing Manager             2  273       274        North American Sales Manager  2  273       285        Pacific Sales Manager         2  16        23         Marketing Specialist          3  274       275        Sales Representative          3  274       276        Sales Representative          3  285       286        Sales Representative          3 | |