Difference between CTE and Temp Table and Table Variable

*Posted By : Shailendra Chauhan, 15 Sep 2013*

*Updated On : 04 Oct 2013*

*Total Views : 124,813*

*Keywords : cte vs temp table vs table variable, local temp table vs global temp table, when to use cte and temp table and table variable*

Temp Table or Table variable or CTE are commonly used for storing data temporarily in SQL Server. In this article, you will learn the differences among these three.

CTE

CTE stands for Common Table expressions. It was introduced with SQL Server 2005. It is a temporary result set and typically it may be a result of complex sub-query. Unlike temporary table its life is limited to the current query. It is defined by using WITH statement. CTE improves readability and ease in maintenance of complex queries and sub-queries. Always begin CTE with semicolon.

A sub query without CTE is given below :

1. **SELECT \* FROM (**
2. **SELECT Addr.Address, Emp.Name, Emp.Age From Address Addr**
3. **Inner join Employee Emp on Emp.EID = Addr.EID) Temp**
4. **WHERE Temp.Age > 50**
5. **ORDER BY Temp.NAME**

By using CTE above query can be re-written as follows :

1. **;With CTE1(Address, Name, Age)*--Column names for CTE, which are optional***
2. **AS**
3. **(**
4. **SELECT Addr.Address, Emp.Name, Emp.Age from Address Addr**
5. **INNER JOIN EMP Emp ON Emp.EID = Addr.EID**
6. **)**
7. **SELECT \* FROM CTE1 *--Using CTE***
8. **WHERE CTE1.Age > 50**
9. **ORDER BY CTE1.NAME**

When to use CTE

1. This is used to store result of a complex sub query for further use.
2. This is also used to create a recursive query.

Recursive Queries Using Common Table Expressions

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.

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

Example :

WITH MenuTbl as

(

SELECT \*

from s\_menu\_mst

where menu\_id = '103N'

UNION ALL

SELECT b.\*

from s\_menu\_mst as b

inner join MenuTbl m ON m.pmenu\_id = b.menu\_id

)

select \* from MenuTbl

menu\_id pmenu\_id window\_name

|  |  |  |
| --- | --- | --- |
| 103N | MP | BOM SEARCH |
| MP | MT | PRODUCTION |
| MT | TPRO | MATERIAL |
| TPRO | A | PRODUCTION |
| A | ROOT | TRANSACTIONS |

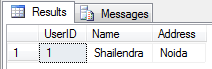
Temporary Tables

In SQL Server, temporary tables are created at run-time and you can do all the operations which you can do on a normal table. These tables are created inside Tempdb database. Based on the scope and behavior temporary tables are of two types as given below-

1. Local Temp Table

Local temp tables are only available to the SQL Server session or connection (means single user) that created the tables. These are automatically deleted when the session that created the tables has been closed. Local temporary table name is stared with single hash ("#") sign.

* 1. **CREATE TABLE #LocalTemp**
  2. **(**
  3. **UserID int,**
  4. **Name varchar(50),**
  5. **Address varchar(150)**
  6. **)**
  7. **GO**
  8. **insert into #LocalTemp values ( 1, 'Shailendra','Noida');**
  9. **GO**
  10. **Select \* from #LocalTemp**

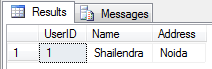


The scope of Local temp table exist to the current session of current user means to the current query window. If you will close the current query window or open a new query window and will try to find above created temp table, it will give you the error.

1. Global Temp Table

Global temp tables are available to all SQL Server sessions or connections (means all the user). These can be created by any SQL Server connection user and these are automatically deleted when all the SQL Server connections have been closed. Global temporary table name is stared with double hash ("##") sign.

* 1. **CREATE TABLE ##GlobalTemp**
  2. **(**
  3. **UserID int,**
  4. **Name varchar(50),**
  5. **Address varchar(150)**
  6. **)**
  7. **GO**
  8. **insert into ##GlobalTemp values ( 1, 'Shailendra','Noida');**
  9. **GO**
  10. **Select \* from ##GlobalTemp**



Global temporary tables are visible to all SQL Server connections while Local temporary tables are visible to only current SQL Server connection.

Table Variable

This acts like a variable and exists for a particular batch of query execution. It gets dropped once it comes out of batch. This is also created in the Tempdb database but not the memory. This also allows you to create primary key, identity at the time of Table variable declaration but not non-clustered index.

1. **GO**
2. **DECLARE @TProduct TABLE**
3. **(**
4. **SNo INT IDENTITY(1,1),**
5. **ProductID INT,**
6. **Qty INT**
7. **)**
8. ***--Insert data to Table variable @Product***
9. **INSERT INTO @TProduct(ProductID,Qty)**
10. **SELECT DISTINCT ProductID, Qty FROM ProductsSales ORDER BY ProductID ASC**
11. ***--Select data***
12. **Select \* from @TProduct**
14. ***--Next batch***
15. **GO**
16. **Select \* from @TProduct *--gives error in next batch***

Note

1. Temp Tables are physically created in the Tempdb database. These tables act as the normal table and also can have constraints, index like normal tables.
2. CTE is a named temporary result set which is used to manipulate the complex sub-queries data. This exists for the scope of statement. This is created in memory rather than Tempdb database. You cannot create any index on CTE.
3. Table Variable acts like a variable and exists for a particular batch of query execution. It gets dropped once it comes out of batch. This is also created in the Tempdb database but not the memory.