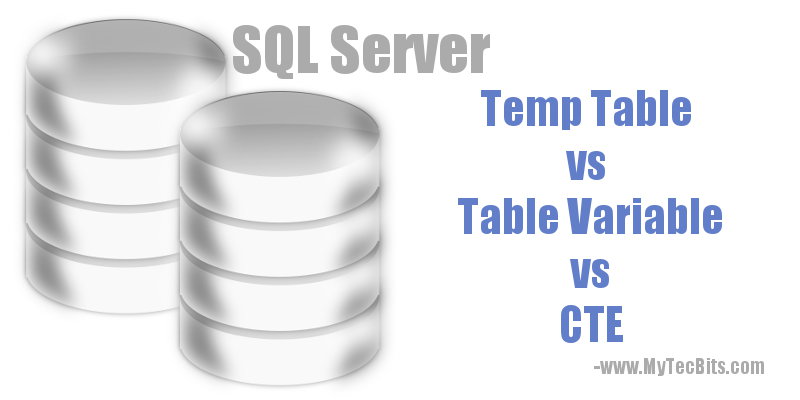
Temp Table Vs Table Variable Vs CTE In SQL Server

[Dahlia Sam](https://www.mytecbits.com/author/dahliatwinkle)   [SQL Server](https://www.mytecbits.com/category/microsoft/sql-server)   Published Mar 2, 2017   Updated Aug 18, 2018   [5 Comments](https://www.mytecbits.com/microsoft/sql-server/temp-table-vs-table-variable-vs-cte#comments)

* [Click to share on Facebook (Opens in new window)](https://www.mytecbits.com/microsoft/sql-server/temp-table-vs-table-variable-vs-cte?share=facebook&nb=1)
* [Click to share on LinkedIn (Opens in new window)](https://www.mytecbits.com/microsoft/sql-server/temp-table-vs-table-variable-vs-cte?share=linkedin&nb=1)
* [Click to share on Twitter (Opens in new window)](https://www.mytecbits.com/microsoft/sql-server/temp-table-vs-table-variable-vs-cte?share=twitter&nb=1)
* [Click to share on Reddit (Opens in new window)](https://www.mytecbits.com/microsoft/sql-server/temp-table-vs-table-variable-vs-cte?share=reddit&nb=1)
* [Click to share on WhatsApp (Opens in new window)](https://api.whatsapp.com/send?text=Temp%20Table%20vs%20Table%20Variable%20vs%20CTE%20in%20SQL%20Server%20https%3A%2F%2Fwww.mytecbits.com%2Fmicrosoft%2Fsql-server%2Ftemp-table-vs-table-variable-vs-cte)
* [Click to share on Pocket (Opens in new window)](https://www.mytecbits.com/microsoft/sql-server/temp-table-vs-table-variable-vs-cte?share=pocket&nb=1)
* [Click to email this to a friend (Opens in new window)](https://www.mytecbits.com/microsoft/sql-server/temp-table-vs-table-variable-vs-cte?share=email&nb=1)
* [Click to print (Opens in new window)](https://www.mytecbits.com/microsoft/sql-server/temp-table-vs-table-variable-vs-cte#print)



I’m often getting questions on when to use the **Temp Table**, **CTE** (Common Table Expression) or the **Table Variable** In SQL Server. So, I thought of writing an article about these three temporary storage units and list down the differences between Temp Table vs Table Variable vs CTE.

Temp Table

Temp tables are otherwise called as **hash tables** or **temporary tables**.The name of the temp table starts with single number sign (#) or double number sign (##). The temporary tables are created during the execution time. As the name suggests, temp tables are available only in the scope where they are created. They are stored in the **TempDB** and will get dropped automatically after the scope. The name of the temp tables can have a maximum of 116 characters. another limitation of temp table compared with their physical counterpart is that they cannot be partitioned.

Temporary tables are of two types, **Local Temporary Tables** and **Global Temporary Tables**. This classification is based on the scope of their existence. Let’s see them in detail.

1. Local Temp Table

The local temp table’s name is prefixed with single number sign (#) (Example: #TableName). These local temp tables are available only in the current session. I.e. if you create a hash table in a stored procedure, then the table will be available only for that stored procedure or any other nested stored procedures called from inside that stored procedure. Once the stored procedure finishes execution, the hash table drops automatically from the TempDB. So only SQL user/connection which created the temp table alone can use it.

Syntax Of Local Temp Table

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | CREATE TABLE #EmployeeLocal  (   EmpID int,   EmpName varchar(100),   EmpAddress varchar(300),   EmpDOB Date  )  GO  INSERT INTO #EmployeeLocal values ( 235, 'John Doe','Minneapolis, MN', '1975-03-01');  GO  SELECT \* FROM #EmployeeLocal |

Usage Of Local Temp Table

You can use local temp tables:

* To store temporary data.
* If the size of the temporary data is huge, say more than 100 rows.
* When the user or connection which creates them alone can use it.
* If you want to use explicit transactions against the temporary data.
* When you may need to create indexes.
* If you may need to apply read lock.
* You **CANNOT** use temp tables in User Defined Functions (UDF).

2. Global Temp Table

The global temp table’s name is prefixed with double number sign (##) (Example: ##TableName). The global temp tables are available for all the sessions or the SQL Server connections. I.e. Multiple SQL Server users can use the same temp table. The table exists till the creates session and all the other sessions or connections using the global temp table closes. Once all the sessions and connections stops using the global temp table, it will automatically drops from the TempDB.

Syntax Of Global Temp Table

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | CREATE TABLE ##EmployeeLocal  (   EmpID int,   EmpName varchar(100),   EmpAddress varchar(300),   EmpDOB Date  )  GO  INSERT INTO ##EmployeeLocal values ( 235, 'John Doe','Minneapolis, MN', '1975-03-01');  GO  SELECT \* FROM ##EmployeeLocal |

Usage Of Global Temp Table

You have to use global temp table, if you want to use the advantages of local temp table plus the feature of multiple users / sessions / connections should have access to the same table.

Table Variable

Table variable is a special kind of data type. It has most of the features of a normal variable along with the capability of storing a result set. This stored result set can be used within a batch. Just like temp table, table variable is also stored in TempDB.

The table variable needs to be declared just like a variable. While declaring, we have to specify the column details.

Syntax Of Table Variable

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | GO  DECLARE @EmployeeLocal TABLE  (   EmpID int,   EmpName varchar(100),   EmpAddress varchar(300),   EmpDOB Date  )    INSERT INTO @EmployeeLocal values ( 235, 'John Doe','Minneapolis, MN', '1975-03-01');    SELECT \* FROM @EmployeeLocal  GO |

Usage Of Table Variable

You have to use table variable only:

* To store temporary data in user defined functions (UDF), stored procedures and query batches. In fact it is the means of returning result set in table-valued user defined functions.
* If the volume of data is less, say less than 100 rows. Microsoft recommends to use Temp Table if you have more than 100 rows of data. Because the optimizer in some cases may ignore the number of records in table variable while generating the query plan.
* When you want to use the temporary data just like referencing a table in SELECT, INSERT, UPDATE and DELETE query.
* If you don’t want to have transactions against the temporary data.
* When you don’t need to alter the table structure after creating it.

CTE (Common Table Expression)

Common Table Expression or CTE is a temporary result set generated from SELECT query defined by WITH clause. The scope of the common table expression is limited to the SELECT, UPDATE, INSERT or DELETE statement which is immediately following it.

There are two types of Common Table Expression **Non-Recursive CTE** and **Recursive CTE**.

1. Non-Recursive CTE

Non-recursive common table expression is the generic form of CTE. It does not have any reference to itself in the CTE definition.

2. Recursive CTE

When a CTE has reference in itself, then it’s called recursive CTE.

Syntax Of CTE

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | ;WITH EmployeeBornBefore1985\_CTE (EppID, EmpName, YearOfBirth)  AS  (      SELECT EppID, EmpName, YEAR(EmpDOB) AS YearOfBirth      FROM EmployeeMaster      WHERE YEAR(EmpDOB) &amp;amp;amp;amp;amp;amp;amp;amp;lt;= 1985  )  SELECT      EppID, EmpName, YearOfBirth  FROM      EmployeeBornBefore1985\_CTE  WHERE      EMPName Like 'C%'  GO |

Usage Of CTE

* CTE is usually used to temporarily store the result set of a complex sub-query, so as you can use the result set later in the main query.

Temp Table Vs Table Variable Vs CTE

| **Sl.#** | **Temp Table** | **Table Variable** | **CTE** |
| --- | --- | --- | --- |
| 1 | Scope wise the local temp table is available only in the current session.  The global temp tables are available for all the sessions or the SQL Server connections. | The scope of the table variable is just within the batch or a view or a stored procedure. | The scope of the CTE is limited to the statement which follows it. |
| 2 | Temp tables are stored in TempDB. | Table variables are also stored in TempDB. | The result set from CTE is not stored anywhere as that are like disposable views. |
| 3 | The name of the temp table can have only up to 116 characters. | The name of the table variable can have up to 128 characters. | Not Applicable |
| 4 | Considering the performance, it is recommenced to use temp table for storing huge data, say more than 100 rows. | Table variable is recommended for storing below 100 rows. | No such performance consideration. CTE normally used as a replacement for complex sub queries. |
| 5 | The structure of temp table can be altered after creating it. | The structure of table variable cannot be altered. | The definition of CTE cannot be changed during run time. |
| 6 | Can explicitly drop temp tables using DROP statement. | Cannot drop table variable explicitly. | Cannot be dropped. |
| 7 | Cannot be used in User Defined Function (UDF). | Can be used in UDF. | Can be used in UDF. |
| 8 | Temp tables take part in transactions. | Table variables wont take part in transactions. | Not Applicable |
| 9 | Index can be created on temp tables. | Index is not possible on table variables. | CTE cannot be indexed. |
| 10 | Can apply read lock on temp tables. | Locking is not possible in table variables. | Locking is not possible in CTE as well. |
| 11 | Constraints can be created on temp tables except FOREIGN KEY. | PRIMARY KEY, UNIQUE KEY and NULL are the only constraints allowed in table variable. | CTE cannot have constraints. |

Related