SQL SERVER Performance Tuning Notes

http://msbiskills.com/

Table Variable – Estimated no of rows = 1 | No Statistics?

SQL Server treats table variable just like it treats any other variable and it estimates them that they have just one row. Even if SQL Server just inserted 1000 rows in them.

According to http://msdn.microsoft.com/en-us/library/ms175010.aspx, Table variables do not have distribution statistics, they will not trigger recompiles. Therefore, in many cases, the optimizer will build a query plan on the assumption that the table variable has no rows. For this reason, you should be cautious about using a table variable if you expect a larger number of rows (greater than 100). Temp tables are better here.

They came in SQL Server 2000. The original intention was to reduce recompiles. As time passes more and more people supported it and many users use to populate huge number of rows and then join with other tables. Just like what we are doing in our example below.

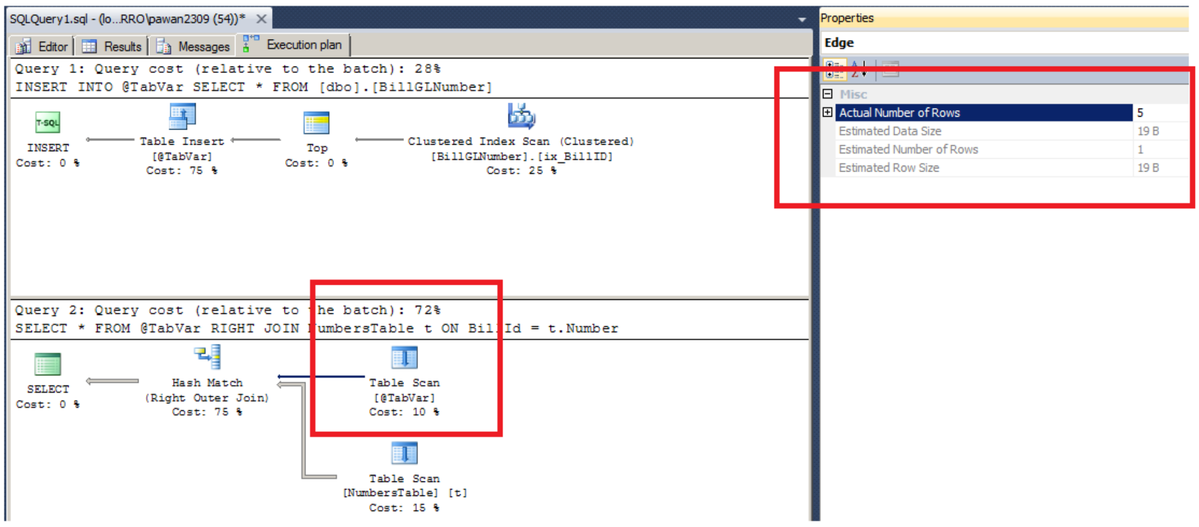
Let's prove with an example

```
DECLARE @TabVar TABLE ( BillId INT , GlNm VARCHAR(1) , Amt INT  )

INSERT INTO @TabVar
SELECT * FROM [dbo].[BillGLNumber]

SELECT * FROM @TabVar
     RIGHT JOIN NumbersTable t ON BillId = t.Number
```

Now let's execute the above batch and check the execution plan. In the below execution plan it is clearly visible that the estimated number of rows and actual number of rows are not matching. Here the number are very small hence doesn't matter much but consider if you have 100K rows.



```
Query 1: Query cost (relative to the batch): 28%
INSERT INTO @TabVar SELECT * FROM [dbo].[BillGLNumber]
```

```
Query 2: Query cost (relative to the batch): 72%
SELECT * FROM @TabVar RIGHT JOIN NumbersTable t ON BillId = t.Number
```

Actual Number of Rows	5
Estimated Data Size	19 B
Estimated Number of Rows	1
Estimated Row Size	19 B

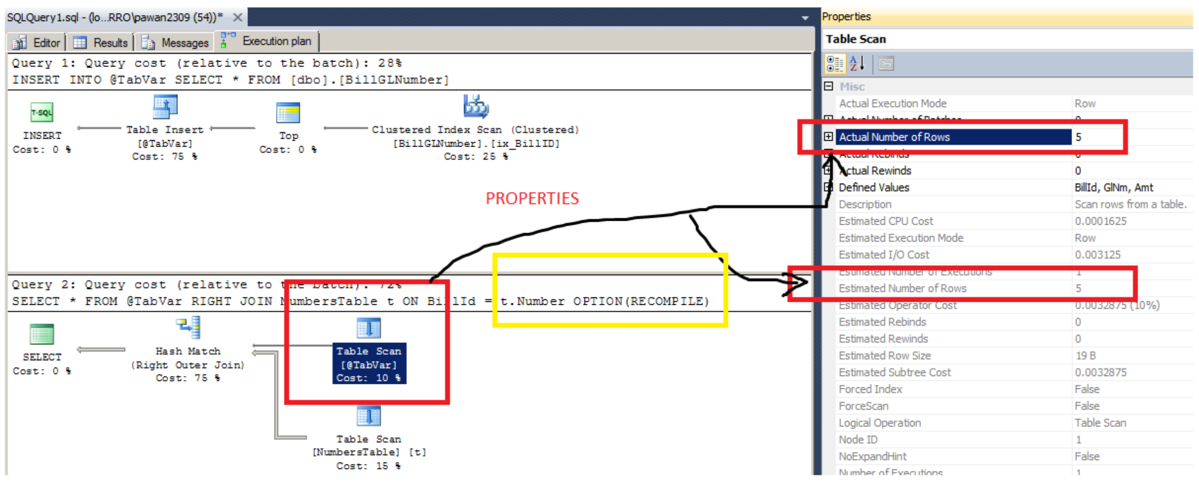
Now let’s add a small variation to the above by adding option recompile.

```
DECLARE @TabVar TABLE ( BillId INT , GlNm VARCHAR(1) , Amt INT  )

INSERT INTO @TabVar
SELECT * FROM [dbo].[BillGLNumber]

SELECT * FROM @TabVar
      RIGHT JOIN NumbersTable t ON BillId = t.Number
OPTION(RECOMPILE)
```

Execution plan analysis of the above query is –



```
Query 1: Query cost (relative to the batch): 28%
INSERT INTO @TabVar SELECT * FROM [dbo].[BillGLNumber]
```

```
Query 2: Query cost (relative to the batch): 72%
SELECT * FROM @TabVar RIGHT JOIN NumbersTable t ON BillId = t.Number OPTION(RECOMPILE)
```

Actual Execution Mode	Row
Actual Number of Batches	0
Actual Number of Rows	5
Actual Rebinds	0
Actual Rewinds	0
Defined Values	BillId, GlNm, Amt
Description	Scan rows from a table.
Estimated CPU Cost	0.0001625
Estimated Execution Mode	Row
Estimated I/O Cost	0.003125
Estimated Number of Executions	1
Estimated Number of Rows	5
Estimated Operator Cost	0.0032875 (10%)
Estimated Rebinds	0
Estimated Rewinds	0
Estimated Row Size	19 B
Estimated Subtree Cost	0.0032875
Forced Index	False
ForceScan	False
Logical Operation	Table Scan
Node ID	1
NoExpandHint	False
Number of Executions	1

Okay so this time our estimates are bang on target and we have same estimated number of rows and actual number of rows.

Please note that SQL Server does not support statistics or track number of rows in a table variable while compiling a query plan.

In SQL Server 2012 a new trace flag 2453 which can help when you are inserted large number of rows in a table variable and joins with other tables. So if this flag is enabled then SQL Server recompiles the statements where table variable is used. And now once the SQL Server knows the row count it can produce a better rather than a sub optimal plan in previous cases.

Well the good news is SQL Server has documented this. The URL is - http://support.microsoft.com/kb/2952444

The article says below things-

This trace flag differs from OPTION (RECOMPILE) in two main aspects.

(1) It uses the same row count threshold as other tables. The query does not need to be compiled for every execution unlike OPTION (RECOMPILE). It would trigger recompile only when the row count change exceeds the predefined threshold.
(2) OPTION (RECOMPILE) forces the query to peek parameters and optimize the query for them. This trace flag does not force parameter peeking.

Note this trace flag must be ON at runtime. You cannot use this trace flag with QUERYTRACEON. This trace flag must be used with caution because it can increase number of query recompiles which could cost more than savings from better query optimization.

Since I don't have SQL 2012 SP2 can't give you practical example. Please follow this link for detailed example –

http://blogs.msdn.com/b/psssql/archive/2014/08/11/if-you-have-queries-that-use-table-variables-sql-server-2012-sp2-can-help.aspx

Thanks for reading!

Pawan Kumar Khowal

MSBISkills.com