Volume 7, Issue 5, May 2017 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Impact of Temporary Tables and Table Variables on Query Performance in SQL Server 2012

Mrityunjay Kumar Gaurav*
Team Lead, Software Engineer, Indus Valley Partners, MCA, IIPS, Devi Ahilya University, Indore, Madhya Pradesh, India

DOI: 10.23956/ijarcsse/SV7I5/0257

Abstract— Query processing is retrieval of information from a database according to a set of selection criteria. Ease of Use, development and troubleshooting tools has made SQL Server popular among developers. Performance of the database systems depends on the query processing. In small size databases performance impact is minimal. Whereas in huge enterprise databases query optimization becomes important. In such cases, onus of writing optimized sql queries lies with the developers, to write best optimized queries considering data volume and runtime join complexities. This paper gives an insight on performance of SQL Table variables and Temporary tables in huge enterprise databases.

Keywords— SQL Server, Performance Analysis, Temporary Tables, Table Variables, Performance Analysis

I. INTRODUCTION

In today's world of modern information technology, which produces powerful computers, it is possible today to collect, store, transfer, and combine huge amounts of data at very low cost. This has enabled companies to have huge databases. However, exploiting the information contained in these databases in an intelligent way turns out to be fairly difficult and it becomes harder to work with these data when it starts to grow humongous. It will be almost impossible to handle or access this large amount of data if we don't do all our database operations in optimized way. In this paper we will compare Table Variables and Temporary Tables and will also analyse the performance of them on large database.

II. TEMPORARY TABLES

Temporary tables are used most often to provide workspace for the intermediate results while processing data. Temp Tables have below salient features:

A. Scope

Within a connection, a temporary table object is visible to the creating level and inner levels (nested). For instance, create a stored procedure and declare a temporary table object within it. Call another stored procedure from first stored procedure (a nested stored procedure) and perform operations like inserting, updating and deleting that temporary table object. Once the main creating level terminates, the temp table is automatically destroyed. It is highly recommended to manually drop the temporary table.

B. Locking

The prospect of table locking is reduced when it comes to local temporary tables since this table is being used by only one user. One aspect which should be kept in mind is that if you cancel a transaction which contains the creation of a temp table object and then cancel that query, an exclusive and update lock can appear on the tempdb. This lock will persist till the complete transaction has closed with a Commit or a RollBack.

C. Transaction

When using a temporary table, a temporary table is an integral part of an outer transaction and therefore, Rollbacks must be supported by Logging

D. Indexing

Indexes on temporary tables can be created explicitly. Hence, there is scope for performance enhancement in temp tables.

E. Constraints

All constraints are available for exploiting on a temp table except when it comes to referring a Foreign Key Constraint .

F. Statistics

SQL Server can create Statistics for temp tables just like we do for permanent tables and therefore, the query optimizer has the option of choosing different plans. Hence, with this in mind, be aware of the scope of Stored Procedure Recompiles.

III. TABLE VARIABLES

Table variables are alternative to temporary tables. Tables are used most often to provide workspace for the intermediate results while processing data. Temp Tables have below salient features:

A. Scope:

Just like any other variable, a Table Variable's scope exists only within the context of the current level. Therefore, unlike Temp Tables, it is not accessible to sub levels (of Stored Procedures)

B. Locking:

Table Variables are not bound to any transaction as they are just like any other variable.

C. Transactions:

Does not participate in any transaction.

D. Indexing

Table Variables can only have indexes that are automatically created with Primary key and Unique Constraints as part of the "Declare" statement.

E. Constraints

Table Variables supports PrimaryKey, Unique Constraint, NULL Constraint, Check Constraint. But they must be incorporated with the creation of the table in the "Declare" statement. Foreign Key Constraint is not allowed.

F. Statistics

Optimizer cannot create any statistics on columns, so it treats table variable has having 1 record when creating execution plans.

IV. COMPARISON

The comparison test was performed on a Windows 2012 R2 64-bit machine running on an Intel Xeon processor X5680 and 16GB of RAM. The SQL Server 2012 Enterprise Edition was used as Database. A testing scenario of modern day e-commerce retail online shopping application was created. The data was collected over time and the response time of various test cases were recorded. Testing methodology is explained as under:

A. DB Model:

A sample real world scenario of online Retail Shopping Application was created. Analysis of Order Management was carried out and below tables were created to store Order data:

i. OrderMaster: To store Order Master data and latest state.
ii. OrderArchive: To store the Order various state changes like – booking, confirmation, shipping, cancellation etc and maintain the complete audit trail of such state changes.

B. Test Data Preparation:

After creating the tables in database, data for around 50,000 orders were inserted in OrderMaster table. For every order, multiple order state changes and order audit data were prepared and inserted in OrderArchive table. To simulate enterprise database, around 5,000,000 rows were inserted in OrderArchive table to maintain the audit and state changes for various orders.

C. Test Case:

We tried to identify the state change trail or complete audit trail for a set of orders. The orders were stored in Temporary Tables and Table Variables. These tables were used to JOIN with archive table to obtain the desired result.

We further ran the queries in samples of 1, 10, 50, 100, 200, 500, 1000 and 2000 orders and analysed the response time as explained below:

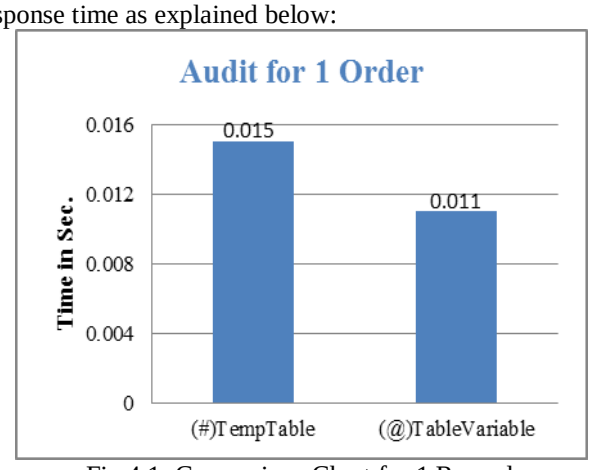

Fig 4.1: Comparison Chart for 1 Record.

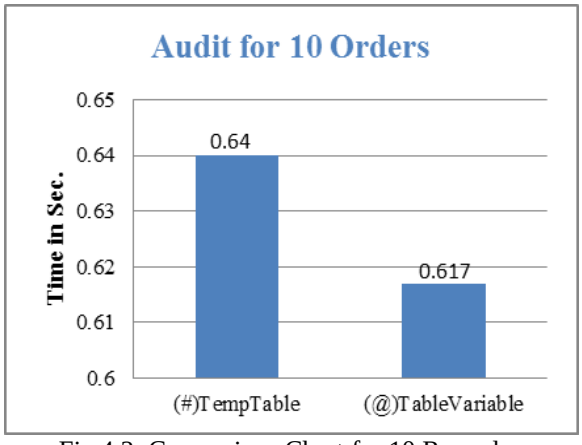

Fig 4.2: Comparison Chart for 10 Records.

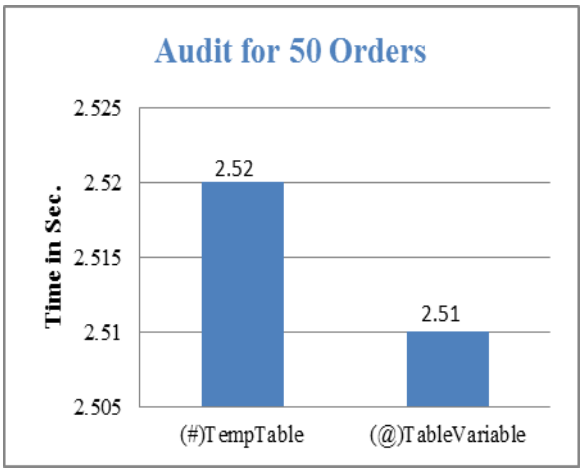

Fig 4.3: Comparison Chart for 50 Records.

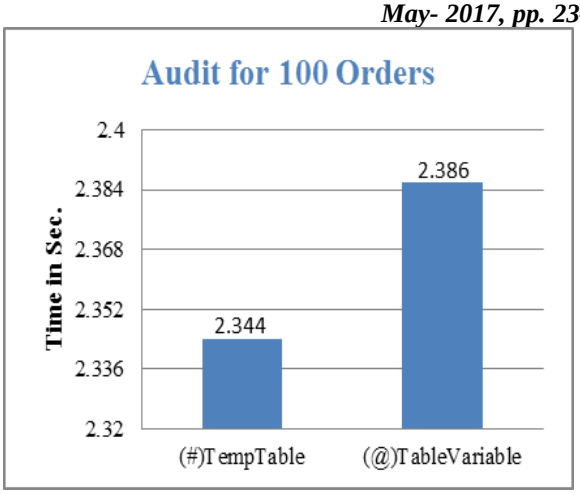

Fig 4.4: Comparison Chart for 100 Records.

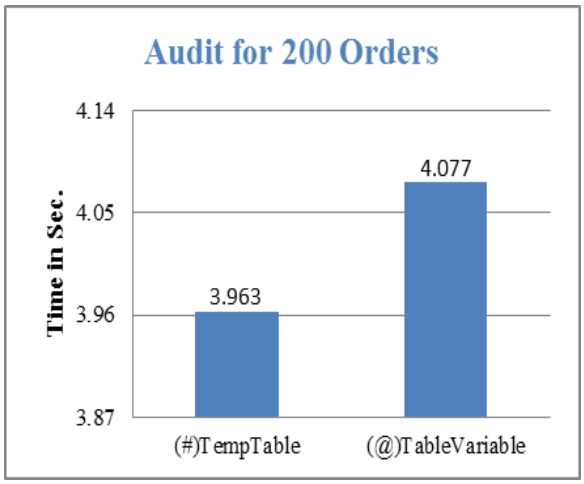

Fig 4.5: Comparison Chart for 200 Records.

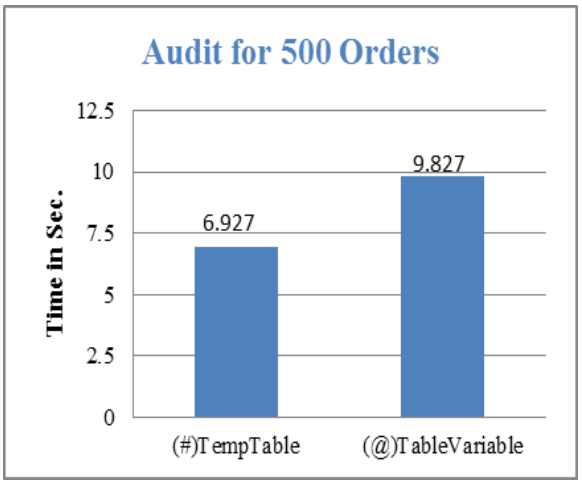

Fig 4.6: Comparison Chart for 500 Records.

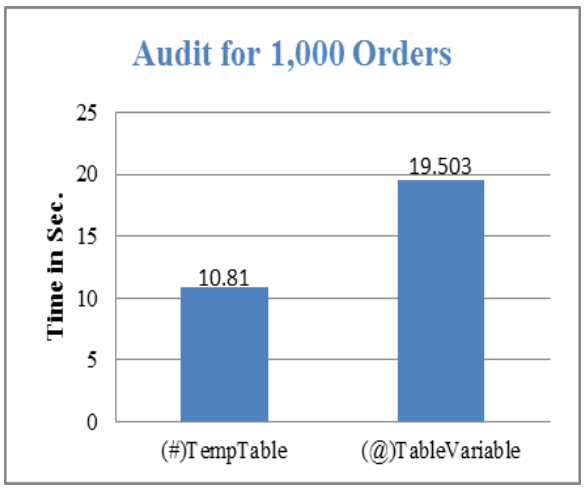

Fig 4.7: Comparison Chart for 1,000 Records.

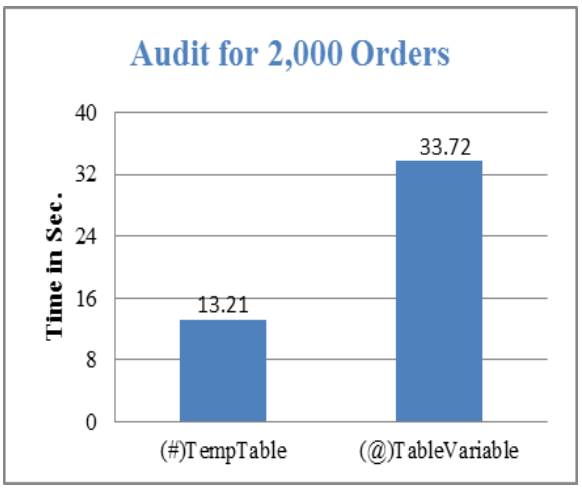

Fig 4.8: Comparison Chart for 2,000 Records.

V. CONCLUSIONS

The purpose of this paper is to analyze the performance of two popular intermediate data storage techniques – Temporary Tables (#Tables) and Table Variables (@Table Variables). In terms of time taken to return results in JOIN Queries in normal conditions, Temporary Tables showed better performance as compared to Table Variables. When JOIN is performed on more than 100 rows, Temporary Tables consistently shows better performance. When JOIN is performed on less than 100 rows, the response time was almost similar with Table Variables having an edge by showing better response time. Considering the test results, for huge enterprise databases, for queries where JOIN queries are performed on bigger tables, usage of Temporary Tables is advisable to store intermediate results and use them in JOINs. For further comparison, effect of creating indexes, latest versions of SQL Server can be considered.

REFERENCES

[1] Milan Matejka, Temporary Tables and Temporary Variables, 31 May 2015, https://www.codeproject.com/Articles/996566/Temporary-Tables-and-Temporary-Variables

[2] Tim Ford, Differences between SQL Server temporary tables and table variables, https://www.mssqltips.com/sqlservertip/1556/differences-between-sql-server-temporary-tables-and-table-variables/

[3] Sean McCown, 10 more do's and don'ts for faster SQL queries, http://www.infoworld.com/article/2604472/database/10-more-dos-and-donts-for-faster-sql-queries.html

[4] Difference between temporary table's in Sql Server 2008? https://stackoverflow.com/questions/19996403/difference-between-temporary-tables-in-sql-server-2008.

