



Author: Arlene Rodríguez-Ortiz

Advisor: Dr. Duffany

Electrical and Computer Engineering and Computer Science Department

Abstract

Databases are essential since are designed to stored and organized data that can be easily managed and accessed. They are crucial to many organizations, companies, and are used in many aspects of our lives. The relational database, based on the relational model, and represented in a tabular way, is one of the most used. Relational database management systems are used to maintain them. One of the most known languages for querying and maintaining relational databases is the Structured Query Language. On this paper article, the researcher explored different optimization tools in MS Azure SQL Server Database that could bring information that could help students and developers to optimize their queries and improve performance.

Introduction

This article has the intention to explore different SQL optimization tools for analyze and optimize queries, that might be useful to other professionals, using the Microsoft Azure SQL Database the SQL language. Management systems brings some tools, but there are other external and describe as more robust tools that could be used to address the optimization of queries. The tools that will be used are SentryOne Plan Explorer [1] and dbForge Studio Query Profiler [2].

Background

The focus of this study in the relational database and RDBMS, which has tabular data concept, based on the relational model. Most relational database management systems use the Structured. Query Language (SQL) to access the database, a set-oriented query standardized language. SQL knowledge and skills are prevalent among the field of software engineering, computer science, and information systems [4].

Query optimization is one of the factors that affect application performance. The query optimizer is responsible for the process of cautiously choosing a suitable query plan from a space of possible execution plans. The query optimization module only considers query plans that can be implemented by the DBMS access algorithms (implements relational algebra operations or combinations) and that apply to the specific query, as well as to the specific physical database design [11]. In terms of cost-based queries, you have a reasonable plans before you can fine-tune. Currently, exist some SQL optimization tools that helps optimize SQL queries, and help you see in depth the execution plans. The database administrators or users should examine the plans, and other return cost and time execution.

Problem

The data management is crucial in any field of study. As in many other things, the world of informatics thrives on the collaboration, the share of knowledge and in the support and integration of other resources.

On this study the researcher aims to the path of automatic query optimization without leaving important considerations. Attempts to find working tools that complement and improve databases; thus the applications, that could be integrated in the developer practices.

Methodology

For the experimental evaluation, real life examples of tables, views, and procedures were created. The database objects recollect diverse of operators and complexity to test against the optimization tools. As mentioned, testing data records on every table were created.

The tools were created in terms of what it brings for query analyze and/or in terms of help improved performance. Also, other tools that would help to build or add to the query, were considered. One of the criteria was that the tools must be compatible with MS Azure cloud database. Also, that tool presented additional functionalities apart from those that the SSMS brings.

As an example, to see analysis and have information to measure performance, the same specific query was executed, before and after creating a suggested index, that could help in the performance. The objective was to see what, and which information will be presented in both tools.

To evaluate performance, it was considered the following data metric columns: Duration, CPU, and Reads. To evaluate performance, you must consider all the aspects mentioned together in the execution. The columns were obtain executing Actual plans. The Actual plan shows the real steps of calculation, unlike the estimated execution plan that is based on the statistics that could be outdated, and are stored in the plan cache without the need of execution. To obtain these metrics Actual Plan will be needed.

Results and Discussion

The findings are the following by tool:

First tool : SentryOne Plan Explorer Results

Statement	Est Cost %	Comp...	Duration	CPU	...	Reads	Writes	Est I...	Est Rows	Actual Rows
select * from [dbo].[vwUserOnRole]	100.0%	3	2	2	...	1,046		100...	502	356

Table	LOB...	LOB...	LOB...	LOB P...	LO...	Page...	Rea...	Page S...	Physical Reads	Logical Reads	Scan Count
Workfile	0	0	0	0	0	0	0	0	0	0	0
Worktable	0	0	0	0	0	0	0	0	0	0	0
MOCK_user_role	0	0	0	0	0	0	0	0	0	1,008	502
MOCK_user	0	0	0	0	0	0	0	0	0	36	1
MOCK_role	0	0	0	0	0	0	0	0	0	2	1
Totals	0	0	0	0	0	0	0	0	0	1,046	504

Figure 1: SentryOne Plan Explorer Table I/O results of the query example execution, before the index, on the Actual plan execution

The tool included: Table I/O, (presented in the Figure 1), Plan Diagram, Query Columns, Plan Tree, Join Diagram, Top Operations, and Index Analysis.

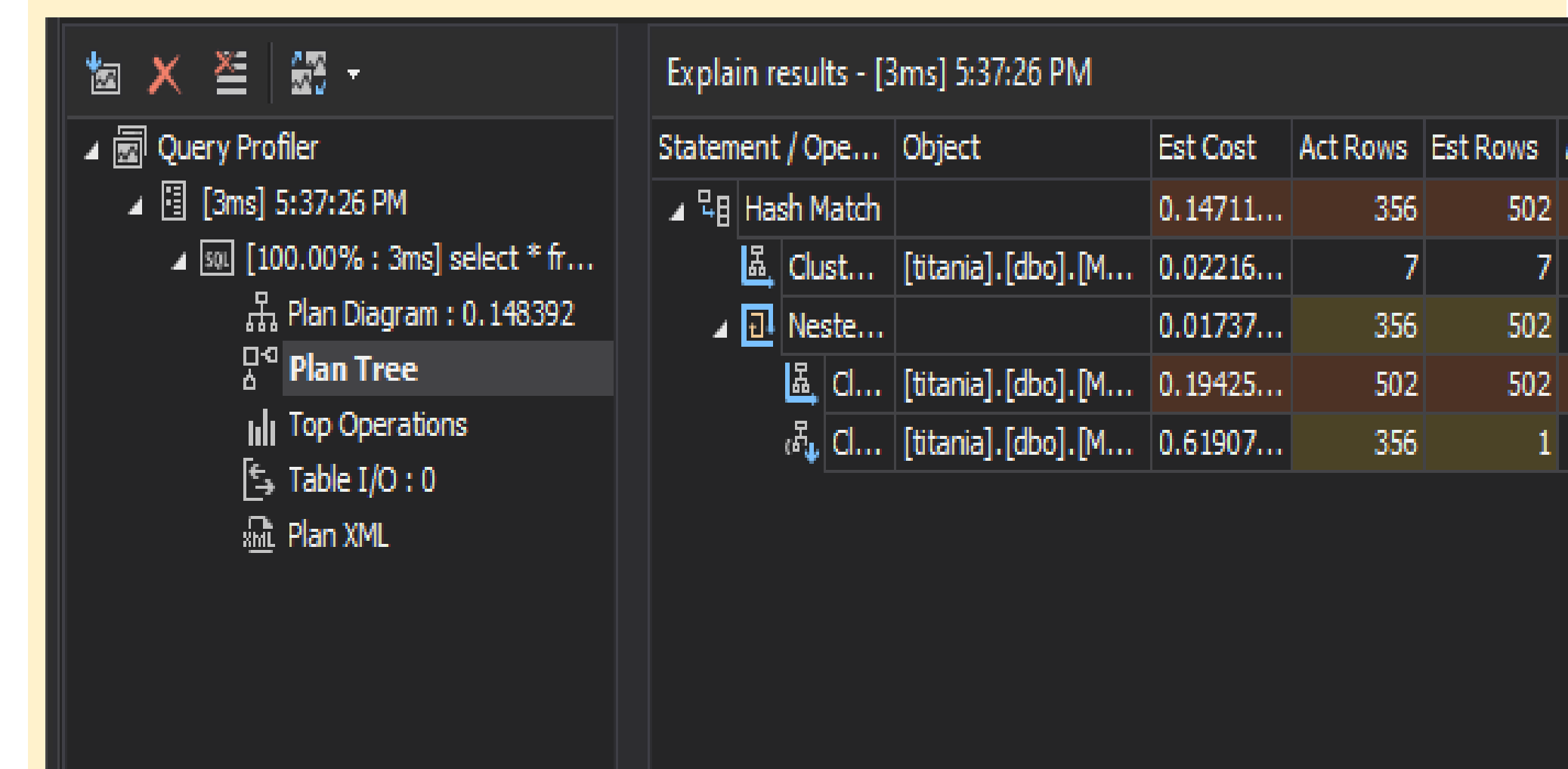
After creating the index, the same query was executed on the Plan Explorer, different results were presented in the total of Logical Reads. The tool presented a way to measure, and some improvement, even though the Duration and CPU remained the same. It helps in the way that it shows that the index reduces the total of logical reads. You would want to have the least number of reads for faster response and better performance; in this context where the Duration and CPU remains the same.

Results and Discussion

- Diagram presented detailed information where can be seen the most expensive operations and optimize the SQL code accordingly.
- The plan tree showed the difference highlighted between the Estimation and Actual rows. This is important because it could indicate a problem with statistics for one or more tables/indexes in the query that could be updated.
- It also included the Index Analysis, which is divided by nodes. It shows table column information: density, last statistics updates, an option to update statistics, Avg Length, Estimated Size, Predict etc.

Second Tool: dbForge Studio Query Profiler Results

Second tool: Using dbForge Studio Query Profiler. Aspect of analysis were examined by executing the example query, before the index, with the live query profiling mode. The example query was executed, after the index, to see the structure, information, and the results.



Statement / Ope...	Object	Est Cost	Act Rows	Est Rows
Hash Match		0.14711...	356	502
Clust...	[titania].[dbo].[M...	0.02216...	7	7
Neste...		0.01737...	356	502
Cl...	[titania].[dbo].[M...	0.19425...	502	502
Cl...	[titania].[dbo].[M...	0.61907...	356	1

Figure 2: dbForge query profiler: Plan Tree, shows the result of the example query on the live query profiling mode.

The results were presented on this format: Plan Diagram, Plan Tree (Figure 2), Top Operations, Table I/O, Plan XML. The plan tree table shows the Actual Rows, in the diagram "Act Rows", fields found in an Actual Plan.

However, the Table I/O which contains the columns of the metrics (Logical Reads, Scan Count) that were necessary, did not bring any information. Due to this lack of information, different result in those fields could not be compared and analyzed. In other cases, some results were contemplated, since is part of the structure. But after trying several queries it did not show. This could be due to there is not a full Actual Plan available.

- The Plan diagram presented, as well, detailed information where can be analyze the most expensive operation or could be useful to investigate why the query optimizer chose one plan to another.
- Additionally, even though there was not part of the query profiler and not an automatic tool for query reformulation there was very useful tool, a query builder by Devart dbForge, that helped build complex SQL queries through visual interface without manual. It simplifies the development of SQL queries and could help students and users who often create database queries.

Conclusions

The tools were very helpful in demonstrating how queries, as developers, impacts the performance; in showing valuable information to detect inefficiencies and to have better practices and participation on this task. Both presented Plan Diagrams, Plan Trees (that gave attention by highlighting discrepancies, high cost, or aspect that required attention), Top Operations, among other features.

The first tool seems to be effective in showing how a query could improve performance. It also gives more information respecting the structure and the columns on the Table I/O, and has extra columns regarding the page server reads. It seems to work on giving results as well. The second tool, besides the example used in the previous section along with other queries, did not presented results on the table I/O. Also, the first tool gave an Index Analysis, that help the tools of performance by updating statistics directly. This could be investigated in detailed in a future investigation.

The study demonstrates that SQL performance tuning can make use of variety of techniques and tools together. The tools for query tuning are complex, and mostly show through processes and statistics that need to be studied.

Future Work

It would be relevant to keep using these tools and give more documentation on real scenarios, that could help other studies in demonstrate the possibilities of these tools and utilize them in other spaces, like the academic area. Furthermore, innovation, automation, and precision in the management of plan execution, and education on this matter. For future work it would be interesting to see more accessible tools, more compatible with different environments, and keep using and testing this research results to tried on other investigations.

Acknowledgements

I would like to acknowledge Dr. Jefferey Duffany for guiding me through this process. I would also like to thank all the professors in the Computer Science & Computer Engineering Department for their teaching. I am grateful for each person who with their study and work help and influence others to continue growing in the field of science and as a society.

References

- [1] SolarW inds World Wide, "Plan Explorer," [Online]. Available: <https://www.sentryone.com/plan-explorer?hsCtaTracking=40dc738e-8792-49e1-b2cb-0d3be75bdc75%7Cefa15c6f-a7e2-446a-9879-676bd6629789>. [Accessed 10 May 2021].
- [2] Devart, "SQL Query Plan Tool," [Online]. Available: <https://www.devart.com/dbforge/sql/studio/sql-query-profiler.html>. [Accessed 30 April 2021].
- [3] T. Taipulus and V. Seppanen, "SQL Education: A Systematic Mapping Study and Future Research Agenda," 2020. [Online]. Available: https://www.researchgate.net/publication/342759889_SQL_Education_A_Sytematic_Mapping_Study_and_Future_Research_Agenda. [Accessed 04 April 2021]
- [4] R. Elmasri and S. Navathe, "Fundamentals of database systems," 2007. [Online]. Available: <https://www.auhd.site/ufpfiles/elibrary/Azal2020-01-22-12-28-11-76901.pdf>. [Accessed 09 April 2021].
- [5] S. Chaudhuri, "An Overview of Query Optimization in Relational Systems," 1998. [Online]. [Accessed 22 04 2021].
- [6] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," 1970. [Online]. Available: <https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf>. [Accessed 30 April 2021].