Considerations on SQL Optimization Tools in MS Azure SQL Server Database using SQL language

Arlene Rodríguez-Ortiz
Master's in Engineering, Software Engineering
Dr. Jeffrey Duffany
Electrical and Computer Engineering and Computer Science Department
Polytechnic University of Puerto Rico

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.*

Key Terms *— Query Plans, Query Profiler, SQL Optimization Tools, and Structured query language (SQL).*

INTRODUCTION

Databases are a fundamental component of software and management solutions. The user query them when he searches through weather applications, social media, virtual libraries, e-commerce; are used from hospitals systems to different organizations. The database is broadly defined as "a usually large collection of data organized especially for rapid search and retrieval (as by a computer)" [1]. Most likely the end user access information through interfaces that leans into a one or more storage of data implemented by software developers, frequently a database.

Even though there are other ways to store data for an application, the databases give possibilities of better management. A database management system (DBMS) is a software package designed to define, manipulate, retrieve, and manage data in a database [2]. Also used as an interface between the end user and the database.

There are several DBMS, among them Relational DBMS. The focus of this study in the relational database and RDBMS, which has tabular data concept. It is important to mention the relational model as a starting point when this subject is discussed. The model introduced by Codd (1970), represents uniform data structures in a database as a collection of relations. This topic will be elaborate according to its relevance in the study [3].

Most relational database management systems use the Structured Query Language (SQL) to access the database, a set-oriented query standardized language. SQL remains currently a predominant in database management systems. It has been incorporated into several commercial databases management systems products such as Microsoft SQL Server, Oracle Database, and Postgres, among others. SQL knowledge and skills are prevalent among the field of software engineering, computer science, and information systems [4].

The developer that works with SQL language and databases, must always try to optimize the queries to improve application performance. Management systems brings some tools, but there are other external and describe as more robust tools that could be used to address this practice.

First, advanced query tunning tools of SQL Server plans that could provide information and inquired in database performance will be evaluated. This article has the intention to explore different SQL optimization tools for analyze and optimize queries, that might be useful to other professionals, other than DBAs, developers for example, and students, using the Microsoft Azure SQL Database the SQL language. First, query optimization

should be defined. The tools that will be used are SentryOne Plan Explorer and dbForge Studio Query Profiler.

BACKGROUND

SQL language

SQL was introduced in the early 1970's at IBM. Edgar F. Codd, a mathematician and computer scientist, who worked at the IBM San Jose Research Lab, laid the theorical foundation for relational databases. Relational search procedures, as proposed by him, offered both improved data independence and an enhanced combinatorial freedom for the user [5]. He published a paper, "A Relational Model of Data for Large Shared Data Banks" [6], showing how information was stored in large databases and could be accessed without knowing how the information was structured or where it resided in the database [7]. As mentioned before, in the book "Computer fundamentals and RDBMS", basically relational models represent data as a collection of relations. Relations resembles a table of values, each row in the table represents a collection of related values, which can be interpreted as a fact describing a relationship instance. The table and column names are used to help interpreting the meaning of the values in each row. Commonly, values are of the same data type. In relational database terminology the followings are called: the tuple a row, the attribute a column and the relation a table [3]. A relational database organizes data into sets of interlinked tables [7]. Even though the querying languages could vary on this study the focus will be on SQL.

After studying the relational model of Codd, Donald Chamberlin and Raymond Boyce published the article "Sequel: A Structured English Query Language", which described a detailed technical description of the language syntax [8]. It was originally designed for an experimental relational database system called SYSTEM R. Later been the language for IBM's DB2 and SQL/DS commercial relational DBMSs [3]. Later was shortened to "Structured Query Language" because SEQUEL had already been copyrighted. The publication influences other vendors to implement SQL, and that led to incompatibilities between the different versions. In 1986 the first SQL standard was published (SQL-86), standardized by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO). Since then, has gone through nine revisions, SQL:2019 being the last. Each revision added additional features [8].

The SQL standard specifies the syntax and semantics of two sublanguages: a schema Data Definition Language (DDL) for declaring the structures and integrity constraints (statements include operators like CREATE, ALTER, DROP) and Data Manipulation Language (DML) for declaring the database procedures and executable statements of a specific database application program (statements include SELECT, INSERT, UPDATE, DELETE) [9]. A third unofficial subdivision of SQL commands or sublanguage is the Data Control Language (DCL), focus on data security. These commands determine whether a user can carry out a particular operation or not (such as GRANT, REVOKE). The ANSI group these commands as part of the DDL [10]. Even though the ANSI documentation does not refer to the Transactional Control Language, the commads of COMMIT and ROLLBACK are mention on the Transactions part. These commands are used and mention in the references of MS SQL Server and Oracle, for example [9].

SQL allows users to retrieve, store, modify and delete data. Also create modify and delete database objects (e.g., tables), grant, and revoke user privileges, and group statements in transactions. An SQL command is called a *statement*, and a statement which retrieves data from a database a *query* [4].

Query Optimization

Query optimization is one of the factors that affect application performance. In the context of DBMS, as mentioned in the book, "Fundamentals of Database Systems" [11], a query expressed in SQL (high-level language) must first be scanned, parsed, and validated. The scanner identifies the query tokens (keywords, attribute names, relation names) in the text. The parser checks the query syntax according to the syntax rules and validated by checking that all attribute and relation names are valid and semantically meaningful names in the schema of the database. Later, internal representation of

the query is created, that could be either a tree or a graph data structure (query tree or query graph, respectively). Next, DBMS must devise the query plan or execution strategy for retrieving the results of the query from the database files. Based on this reference, the high-level query steps are represented on Figure 1.

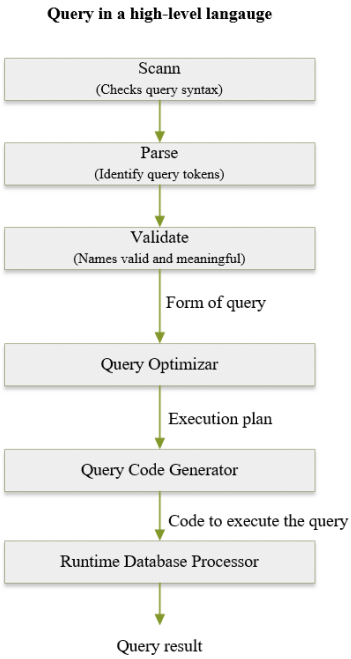

Figure 1

Query in high-level language

Query optimization is the process of cautiously choosing a suitable query plan from a space of possible execution plans. The query block is the basic unit that contains a single SELECT-FROM-WHERE expression, it may include some clauses, example in Figure 2. The query optimizer would choose an execution plan for each query block. The code generator created the code to execute the plan. The runtime database processor has the task of running the query code (complied or interpreted) to produce the query result.

```
SELECT fname, lname, city
FROM user
WHERE city = 'Alameda'
```

Figure 2

Query Block (statement)

Because the optimizer could fail in not choosing the optimal strategy and finding the optimal is commonly time-consuming and could require detailed information, the authors of this book questioned the terminology ("optimization") of this process suggesting a more accurate description like "planning of a good execution strategy".

The high-level query being declarative in nature and it needs the query optimization for queries. A relational DBMS must systematically evaluate alternative query execution strategies and choose a reasonably efficient or near-optimal strategy. 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.

There are two main techniques, usually combine, employed during query optimization: heuristic rules (ruled-based) and systematic estimate cost (cost-base). Since the 1970's there has been work related to query optimization. Pat Selinger, which make a huge contribution to the database industry and became a leading member of the team that built System R, developed a cost-based query optimization that makes work with relational database more cost-effective and efficient. It has been taught in many universities database courses and adopted by most vendors. On an interview in 2008, Selinger [12] elaborates about the cost-based query saying:

"*The trick to cost-based query optimization is estimating a cost for each of the different ways of accessing the data, each of the different ways of joining information from multiple tables and estimating the sizes of results and the savings that you would get by having data in the buffer pools, estimating the number of rows you'll actually touch if you use an index to access the data, and so forth.*

The more deeply you can model the cost of accessing the data, the better the choice you will make for an access path."

The plans can be considered in terms of the optimal and its robustness and having able to have good plans and analyze in the process if the data is deviating from what is expected. She expressed that you need to have

reasonable plans before you can fine-tune. She also thinks that cost-based optimization could continue in the route of automatic query optimization rather than put programmers back into the game of understanding exact data structures and doing the navigation in the application program manually.

Currently, exist some SQL optimization tools that helps optimize SQL queries. Some optimizers that analyze and created different execution plans, besides the optimizer that comes with the RDBMS that can help you to improve your database. The database administrators or users should examine and tune the plans. The plans can also give you the opportunity for indexes that might help to improve performance. To address this matter some databases, provide a plan table which return the cost and time for executing a query [13].

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.

This article has the intention to explore different SQL optimization tools for analyze and optimize queries, using the Azure SQL Server database, with the SQL language, that might be useful. The tools that will be used are SentryOne Plan Explorer and dbForge Studio Query Profiler.

EQUIPMENT AND MATERIAL

Database

Azure SQL Database is an intelligent, scalable, relational database service built for the cloud. The Microsoft Azure SQL Database is "a fully managed platform as a service (PaaS) database engine that handles most of the database management functions such as upgrading, patching, backups, and monitoring without user involvement. Azure SQL Database is always running on the latest stable version of the SQL Server database engine and patched OS with 99.99% availability" [14].

The database is query through the Azure Data Studio with the following descriptions:
Version: 1.28.0 (user setup)
Date: 2021-04-15T00:24:15.710Z
VS Code: 1.51.0
Electron: 9.4.3
Chrome: 83.0.4103.122
Node.js: 12.14.1
V8: 8.3.110.13-electron.0
OS: Windows_NT x64 10.0.19041

RDBMS

The query plans are consulted on the Microsoft SQL Server Management with the following descriptions:
SQL Server Management Studio
15.0.18338.0
SQL Server Management Objects (SMO)
16.100.41011.9
Microsoft Analysis Services Client Tools
15.0.19205.0
Microsoft Data Access Components (MDAC)
10.0.19041.1
Microsoft MSXML
3.0 6.0
Microsoft .NET Framework
4.0.30319.42000
Operating System
10.0.19041

Database Description, Schema and Data

The database design and structure are based in some components of a working web application database. It was made completely in space just for testing. It does not exposed fields or scripts. It is an Azure SQL database with no elastic pool. Service Tier S0, DTUs:10, Included Storage: 250 GB, Maximum Storage: 250 GB.

The database does not contain confidential and sensitive data. We design the fields; add and adjust datatypes through the Mackaroo [15] web application, a free test data generator and API mocking.

Query Optimization Tools

- **dbForge Studio Query Profiler:** The query profiler a visual tool for tuning query performance. The execution diagrams show the slow-executing nodes. Each query in the batch is analyzed is displayed with the cost of each query as a percentage of total batch cost. It has Wait Stats, displays explain results, top operations list, and table I/O. It indicates positions were adding an index to a table, optimizing tables joining, etc., could increase SELECT performance. Also, profiling results history and comparison [16].
- **SentryOne Plan Explorer:** Plan Explorer is a single installation file containing the application and the SSMS add-in which give the option of directly go from SSMS to the tool. In terms of speed, it helps you to the root of query tuning issues by pointing the issues that might exist in the execution plan. This includes highlighting from expensive operations, to allowing you to sort by any metric in most grids. Plan Explorer exposes many details about plan operators. Regarding visibility, with Actual execution plan generation, you can see runtime metrics without manually setting statistics options or digging into a plan's properties [17].

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:

SentryOne Plan Explorer

First tool: The Plan Explorer of SentryOne was easy to install and well documented. The example query was executed, before the index, to see the structure, information, and the results.

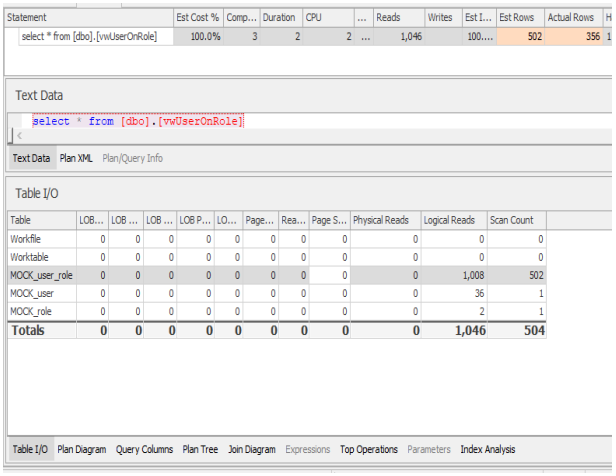

Figure 3

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 3), 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, to compare the results.

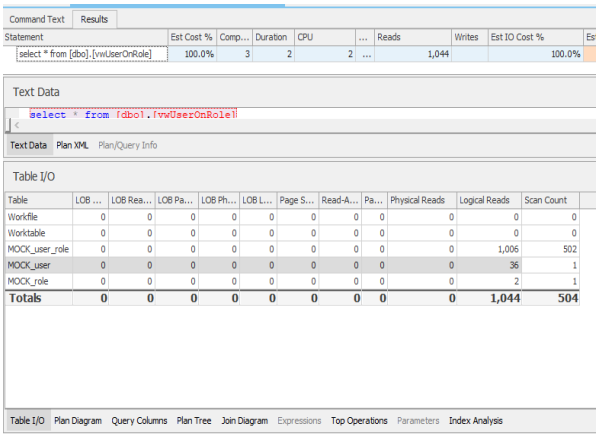

Figure 4

SentryOne Plan Explorer Table I/O results of the query example execution, after the index, on the Actual plan execution

Different results were presented in the total of Logical Reads (Figure 4). 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. There is significance in the context since there could be an exception, were creating an index could cause more logical reads, however a significance reduction of CPU and Duration, and still improved.

In terms of analysis, the Plan 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.

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.

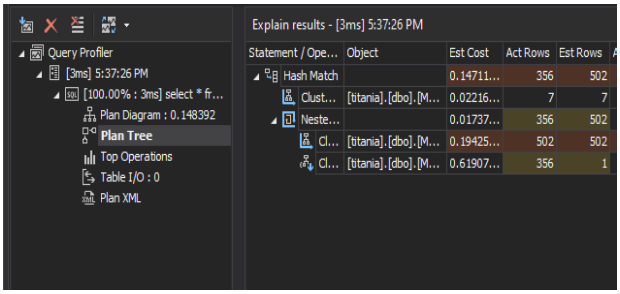

Figure 5

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 5), 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 (Figure 6).

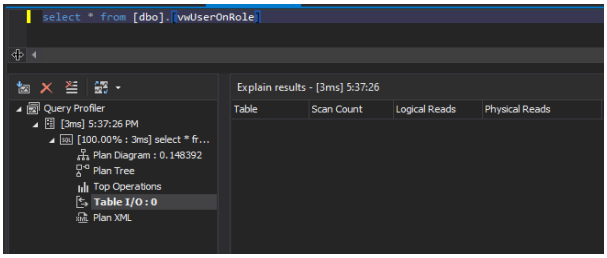

Figure 6

dbForge query profiler: Table I/O with columns like (Logical Reads, Scan Count)

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 (selects and an update) 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.

CONCLUSION

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 tunning are complex, and mostly show through processes and statistics that need to be studied. There should be farther study on this tool. The researcher considered there is more to explore through other sources and with these providers of database performance monitoring and solutions.

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.

Also, there were some limitations trying other automatic optimization tools either due to problems with compatibility with cloud systems and types of databases, administrative and validation problems or because they were on the beta stage. The automatic query tools are limited in suggesting mostly indexes.

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.

REFERENCES

[1] "Merriam-Webster," MERRIAM-WEBSTER DICTIONARIES, [Online]. Available: https://www.merriam-webster.com/dictionary/database. [Accessed 05 March 2021].

[2] "techopedia," [Online]. Available: https://www.techopedia.com/definition/24361/database-management-systems-dbms. [Accessed 30 April 2021].

[3] S. Vaze and S. Joshi, Computer Fundamentals and RDBMS, Global Media, 2009.

[4] 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_Systematic_Mapping_Study_and_Future_Research_Agenda. [Accessed 04 April 2021].

[5] D. Gugerli, "The world as databse: on the relation of software developmnet, query methods, and interpretative independence," 2012. [Online]. Available: https://www.researchgate.net/publication/314438214_The_World_as_Database_On_the_Relation_of_Software_Development_Query_Methods_and_Interpretative_Independence. [Accessed 07 May 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].

[7] K. Wickett and A. Thomer, "Relational data paradigms: What do we learn by taking the materiality of databases seriously?," 2020. [Online]. Available: https://journals.sagepub.com/doi/10.1177/2053951720934838. [Accessed 30 April 2021].

[8] D. D. Cahmberlin, "Early History of SQL," 2012. [Online]. Available: https://web.archive.org/web/20160304073050/http://dbis-informatik.uibk.ac.at/files/ext/lehre/ss11/vo-ndbm/lit/ORel-SQL1999-IBM-Nelson-Mattos.pdf. [Accessed 19 April 2021].

[9] American National Standard Institute, "American National Standard for Information Systems-Database Language-SQL," 1986. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub127.pdf. [Accessed 02 March 2021].

[10] A. I. Din, Structured Query Language (SQL): a Practical Introduction, Blackwell Pub, 1994.

[11] R. Elmasri and S. Navathe, "Fundamentals of database systems," 2007. [Online]. Available: https://www.auhd.site/upfiles/elibrary/Azal2020-01-22-12-28-11-76901.pdf. [Accessed 09 April 2021].

[12] "Database Dialogue with Pat Selinger," December 2008. [Online]. Available: https://cacm.acm.org/magazines/2008/12/3355-database-dialogue-with-pat-selinger/fulltext. [Accessed 03 May 2021].

[13] S. Chaudhuri, "An Overview of Query Optimization in Relational Systems," 1998. [Online]. [Accessed 22 04 2021].

[14] S. S. e. al, "Microsoft Documentation," Microsoft Corporation, 21 09 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview. [Accessed 30 April 2021].

[15] M. Brocato, "mockaroo," Mockraoo, LLC., [Online]. Available: https://www.mockaroo.com/. [Accessed 08 May 2021].

[16] Devart, "SQL Query Plan Tool," [Online]. Available: https://www.devart.com/dbforge/sql/studio/sql-query-profiler.html. [Accessed 30 April 2021].

[17] "IBM Documentation- Query Optimization," IBM Corporation, 2014. [Online]. Available: https://www.ibm.com/docs/en/db2/11.1?topic=performance-query-optimization. [Accessed 05 March 2020].

[18] 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].

[19] A. Yaseen, "SQL Shack SQL Server Estimated Vs Actual Execution Plans," 29 December 2016. [Online]. Available: https://www.sqlshack.com/sql-server-estimated-vs-actual-execution-plans/. [Accessed 1 May 2021].