Journal of Theoretical and Applied Information Technology
10th June 2014. Vol. 64 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EVALUATION SUBQUERY METHODS IN MICROSOFT SQL SERVER 2008

Tanty Oktavia
Bina Nusantara University, School of Information Systems, Jakarta – 11480, Indonesia
E-mail: tanty_oktavia@yahoo.com

ABSTRACT

Frequently user compiles a query to satisfy business process needs, whether directly using DBMS or connecting into application system. There are many methods that can be used to generate desired results to support transaction, but all the query processing should be run effectively and efficiently. A significant aspect of query processing is how to choose an efficient execution strategy to minimize resource usage. Based on competitive environment in industry, many companies compete with each other to be a number one. For this reason, company doing a lot of experiments, in order to accomplish a visions and mission objectives. One of those is increasing performance of the system to support daily activities. Nowadays, most of business process in company already integrated with information technology. All of data is consolidated by database, so user easily doing their job. According to this fact, user does not matter about how many transactions to be process per day, but how much time they need to process that transaction is the priority concern. Because of that, query optimization techniques become more important to be applied in many applications. In this research focused on measurement effectiveness of the method sub query which can be applied to reach optimal execution and present a comparative study of various cost to declare Sub query. This study based on Microsoft SQL Server 2008 platforms.
Keywords: *Query Processing, Microsoft SQL Server 2008, Sub query, Cost Control*

1. INTRODUCTION

Almost all of the business applications require database for the purpose of integration data as well as data distribution among the system. Database Management System (DBMS) as software which manage data should be administered and optimized for better performance. This system should fast respond any kind of possible threats that may be frighten. The performance of database system is influenced by several factors i.e.: database size which growing proportional with the data, increased user base, increase in the user processes, improperly and un-tuned DBMS. All these factors degrade the system response time that would anticipate the performance degradation [1]. This increased load has to be minimizing to stabilize the response rate of system.

One of the major critical issues often happened in a company was inadequate performance of queries to perform the suitable output. Many factors that cause this occurrence, one of them are query processing problem. Since then, a significant amount of research and observation has been done to find an efficient solution for processing queries. A query may be expensive in terms of cost of execution if it is not optimized well [2]. For a long time this matter can be a negative impact for a company because decreasing business performance. The detection of performance degradation is detected by continuously monitoring system performance parameter.

In first generation structure database systems, the low level procedural query language is generally embedded in a high level programming language and the programmer's should select the most appropriate execution strategy. In contrast, with declarative languages such as SQL, the user specifies what data is required rather than how it is to retrieved [3]. This pattern transforms the user responsibility to determine, or even know what method to support good execution strategy. The most important objectives to be considered in order to improve the performance of DBMS are: designing an efficient data schema, optimizing indexes, analyzing execution plans, monitoring access to data, and optimizing query [4]. For this research focused on optimizing query in Microsoft SQL Server 2008 platform.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Microsoft SQL Server 2008 is one of the most widely large scale databases in commercial. As a Database Management System (DBMS), it is a software product that function is to store and retrieve data as complied by software applications. This version aims to make data management self tuning, self organizing, and self maintaining, also provide near zero downtime.

Table 1. Rule Based Optimization Rankings [3]

Rank	Access Path
1	Single row by ROWID (row identifier)
2	Single row by cluster join
3	Single row by hash cluster key with unique or primary key
4	Single row by unique or primary key
5	Cluster join
6	Hash cluster key
7	Indexed cluster key
8	Composite key
9	Single-column indexes
10	Bounded range search on indexed columns
11	Unbounded range search on indexed columns
12	Sort-merge join
13	MAX or MIN of indexed column
14	ORDER BY on indexed columns
15	Full table scan

2. MATERIALS & METHODS

Generally, this study tries to compare the total execution time of all individual operation in order to enhance query performance using sub query methods. A large variety of processing techniques are supported by Microsoft SQL Server 2008 in its SQL Processing engine. For the analysis, the query execution plan and response time are considered. This study is iterative process to measure the impact of sub query methods and reassessing the changing from the query construction from the other method to determine satisfactory performance, but need to realize the performance depends on the amount of data and users' activities that access the application. There are still many factors that should be considered in order to increase optimal performance of queries that connect with application [5]. High performance can be reached by constructing an efficient code at the application level and design the database using suitable techniques. Nevertheless, using superior specification of hardware is one of the performances influenced.

3. RESULT AND DISCUSSION

Information system application is growing faster in the future and the database is going to be larger than before to support business process. Day by day, data is manipulated by operation insert, update, delete this condition makes the system running slow and needs more time to execute the transaction. Depending on application requirements and user needs are force to retrieve the records of a file on either fast or sequential. Disk devices can store records in some logical sequence. Assume one file consists of many thousands or even a million records and user want to retrieve a single records based on a particularly criteria [3]. The disk device may be capable of going directly into the middle of a file to show a record, but to accomplish that, the system need time execution to generate that record with many procedures compilation in the system DBMS. To solve or prevent this indication problem, the database tunings can be the best solution. In this research, the method of database tuning that being applied is the query optimization process on sub query functions, which should optimized for better performance.

The process of DBMS manages query, consist of several stages are as follow: After receiving query from external level, query parser checking semantically and syntax. If there are violation of structure, user right, or procedure; an error message will send to user, otherwise query will be translated into internal level in relational algebra expression. Furthermore, query optimizer selects appropriate optimal method to implement relational algebra and finally generate query execution plan [6]. The query plan is compiled code that contains the ordered steps to carry out the query [7]. Identifying an appropriate plan for execution is very important because these queries can be the determinants of effectiveness transaction. Using statistics on tables and indexes, the optimizer predicts the cost of using alternative access methods to resolve a particular query.

Queries in algebra are constructed using operators. Each relational query describes a step by step procedure for computing the desired output, based on the order in which operators are applied [8]. There are many variations of the operations that are included in relational algebra. The five fundamental operations in relational algebra: selection, projection, Cartesian product, union, set difference, join, intersection, and division operations. The selection and projection operations are unary operations, which operate only on one relation. The other operations work on pairs of relations is therefore called binary operations [3].

Journal of Theoretical and Applied Information Technology
10th June 2014. Vol. 64 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

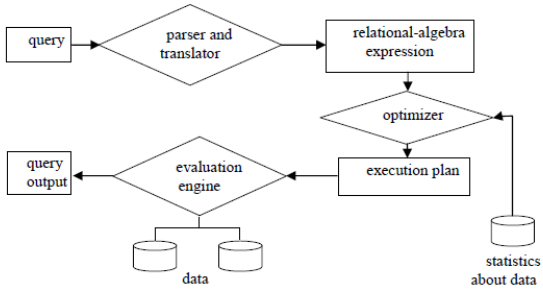

Figure 1 Process Of Query Execution [6]

A Sub query as one of the binary operations, is a query that is nested anywhere inside DML syntaxes, such as INSERT, UPDATE, DELETE, OR SELECT statement. By default, the sub query list only consists of a single column name or expression, except for sub query that use EXISTS keyword. Sub query can be combined with all function of SQL Server 2008. The SQL Server 2008 will resolve the IN condition by accessing the index for a number of times equal to the number of values which to search. There are three types of sub query [3]:

- A scalar sub query returns a single column and a single row.
- A row sub query returns multiple columns, but only a single row.
- A table sub query returns one or more column and multiple rows.

The types of sub query can be applied in accordance what data and how many value of data, a user want to return and satisfy the requirement.

According to the utilization requirements for the system, Microsoft SQL Server 2008 provides indexing method. It divided into clustered index and non clustered index. The clustered indexes are recommended only for tables that are frequently updated. Clustered type indexes are effective when operators like BETWEEN, >, >=, <, <=, <>, !=. Non clustered indexes use is recommended only for databases where updates are infrequent and gives the optimal solution for the "exact match" [5]. Many questions are come up, which attributes are suitable to be applied index to get better performance. Gilenson states there are two sorts of possibilities: primary keys, and search attributes [9].

Indexes are extremely method for searching data, but should keep in mind when the record in a table is modified, the system must take the time to update the table's indexes too. It will do update automatically, but it needs time. If user updates a lot of data, the time that it takes to execute the updates operation and update all the indexes could slow down the operations that are just trying to read the data for applications, and also degrading query response time. Hence that fact, user should beware when applied index in query. The placement of index must be precise with the necessity and procedures.

In transact SQL statements, there is usually no procedure that regulate when to apply a sub query or that does not, because it is not difference between them. User does not concern how to retrieve the data, but how many times the execution occurred. However, in some cases where the data to be returned numerous or the conditions from the query are very complicated, it caused the performance query is going to down. In case of query optimization, it is impractical to search evaluation plans exhaustively, when the optimization of query involves many relations [10].

The table used in the experiment is named *TransactionAttendances*. The *TransactionAttendances* table consists of *TransactionAttendanceId*, *ClassTransactionDetailId*, *BinusianId*, *AttendDate*, *AttendPlace*, *Status*, and *InsertedDate*. The table is populated with 500,000 transactional records. The experiments are performed on ten times with Microsoft SQL Server 2008 platform. The following four queries are used in the experiments for observing time execution. Experiments are performed using both INDEX and NONINDEX. Indexes are applied in ClassTransactionDetailId Ascendingly, BinusianId Ascendingly include InsertedDate; ClassTransactionDetailId Ascendingly, BinusianId Ascendingly include Status, InsertedDate.

Table 2. List Of Query Used For Experiments

No.	Methods	Query
1	IN	SELECT i.ClassTransactionDetailId,i.BinusianId,i.Status FROM Messier.dbo.TransactionAttendances i WHERECONVERT(VARCHAR(50),i.ClassTransactionDetailId)+CONVERT(VARCHAR(50), i.BinusianId)+CONVERT(VARCHAR(50),i.InsertedDate,13) IN(SELECT CONVERT(VARCHAR(50),ClassTransactionDetailId)+CONVERT(VARCHAR(50),

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

		BinusianId)+CONVERT(VARCHAR(50),LastInsertedDate,13) AS [Key] FROM (SELECT ClassTransactionDetailId, BinusianId, MAX(InsertedDate) AS LastInsertedDate FROM TransactionAttendances GROUP BY ClassTransactionDetailId, BinusianId) x) GO
2	EXIST	SELECT i.ClassTransactionDetailId,i.BinusianId,i.Status FROM Messier.dbo.TransactionAttendances i WHERE EXISTS(SELECT 1 FROM (SELECT ClassTransactionDetailId, BinusianId, MAX(InsertedDate) AS LastInsertedDate FROM TransactionAttendances GROUP BY ClassTransactionDetailId, BinusianId) x WHERE CONVERT(VARCHAR(50), i.ClassTransactionDetailId)+CONVERT(VARCHAR(50), i.BinusianId)+CONVERT(VARCHAR(50),i.InsertedDate,13) = CONVERT(VARCHAR(50), ClassTransactionDetailId)+CONVERT(VARCHAR(50), BinusianId)+CONVERT(VARCHAR(50),LastInsertedDate,13)) GO
3	RELATIONAL OPERATOR (=)	SELECT i.ClassTransactionDetailId,i.BinusianId,i.Status FROM Messier.dbo.TransactionAttendances i WHERE CONVERT(VARCHAR(50), i.ClassTransactionDetailId)+CONVERT(VARCHAR(50),i.BinusianId)+CONVERT(VARCHAR(50),i.InsertedDate,13) =(SELECT CONVERT(VARCHAR(50), ClassTransactionDetailId)+CONVERT(VARCHAR(50), BinusianId)+CONVERT(VARCHAR(50),LastInsertedDate,13) AS [Key] FROM (SELECT ClassTransaction DetailId, BinusianId, MAX(InsertedDate) AS LastInsertedDate FROM TransactionAttendances GROUP BY ClassTransactionDetailId, BinusianId) x WHERE x.ClassTransactionDetailId = i.ClassTransactionDetailId AND x.BinusianId = i.BinusianId) GO
4	RELATIONAL OPERATOR (=) + TOP	SELECT i.ClassTransactionDetailId,i.BinusianId,i.Status FROM Messier.dbo.TransactionAttendances i WHERE i.TransactionAttendanceId =(SELECT TOP 1 TransactionAttendanceId FROM TransactionAttendances x WHERE i.ClassTransactionDetailId = x.ClassTransactionDetailId AND i.BinusianId = x.BinusianId ORDER BY x.InsertedDate desc) GO

Based on the above query can be checked, the user want to retrieve the last transaction record that represented by inserted date of each data. First to third query, using an aggregate function that is MAX to return the latest record that user input. For the last query, the experiment try to change the MAX function with TOP, but still refer to the same mean of query and the same amount of result.

Journal of Theoretical and Applied Information Technology
10th June 2014. Vol. 64 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

After doing the experiments, using the same query and same platform, there are results from the experiments on table 3:

Table 3 Time Comparison Of Sub Query Method No Index Applied

Method	NONINDEX (Sec)										Average (Sec)
	1	2	3	4	5	6	7	8	9	10	
IN	3310	2925	3437	2859	2922	2781	3047	2812	2843	3094	3003
EXISTS	2859	2797	2859	3127	2876	3221	4000	4000	2797	2937	3147.3
TOP	3437	3578	3609	3593	3640	3422	3453	3390	3578	3359	3505.9
EQUAL	10359	5031	5047	5562	14813	5126	14187	5344	5678	4875	7602.2

The time execution average shows relational operator sub query methods take the most time to process the query. Relational operators consist of =, <, >, <=, >=, <>, != in WHERE clause, or a HAVING clause. To apply this method enclosed the query in parentheses and combine the relational operators corresponding to the requirements.

IN and EXISTS method does not have a significant differentiation for time processing. On average the comparison between IN and EXISTS only 147.3 second faster if IN applied. Despite the fact IN only return one value, otherwise EXISTS, but IN and EXISTS has no high variance time. This fact can be as a guideline for user to choose the appropriate method, while using sub query method in query application. It depends on how many data, user wants to return in query but for the processes have the same time execution.

Table 4 Time Comparison Of Sub Query Method With Index Applied

Method	INDEX (Sec)										Average (Sec)
	1	2	3	4	5	6	7	8	9	10	
IN	1921	1969	1876	1796	1796	1828	1828	1968	1750	2203	1893.5
EXISTS	2093	1796	1750	2000	1875	1796	1828	2047	1703	1875	1876.3
TOP	78	31	15	33	15	15	15	15	31	15	26.3
EQUAL	203	78	78	62	78	79	49	78	62	62	82.9

To overcome the boundary of existing condition, this study proposed an improved method of query optimization to reduce execution time. The manual query tuning process is analyzed by applying index method that affects the query performance, whereas automatic query tuning process is provided by Microsoft SQL Server 2008. After the indexing is applied, there are extremely transformations of time execution from every activity than before. Average time execution of IN and EXISTS inversely with relational operator. IN and EXISTS need the most time execution, but relational operator and TOP only need the minimal time to execute query. If user decides to use relational operator for sub query or using combination with TOP keyword, INDEXING is the standard requirement that query must be applied to gain the best performance in query execution.

Utilization of IN and EXISTS still have the same levels of time execution. The difference average execution is only 17.2 second faster if IN applied. This condition is contrast with before execution, when index were not applied. There are no significant difference between indexing applied or not for IN and EXISTS keyword. It's up to user to choose what method to be applied in query.

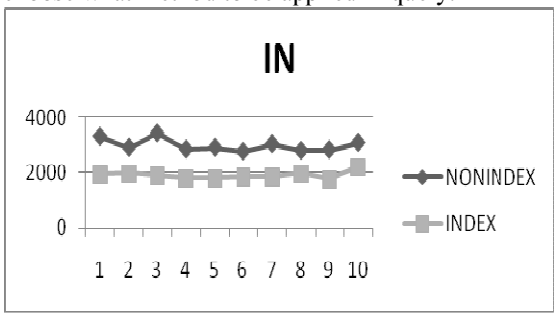

Figure 2 Time Comparison of Query Using IN

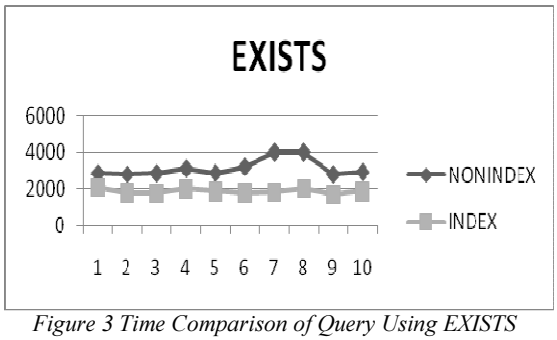

Figure 3 Time Comparison of Query Using EXISTS

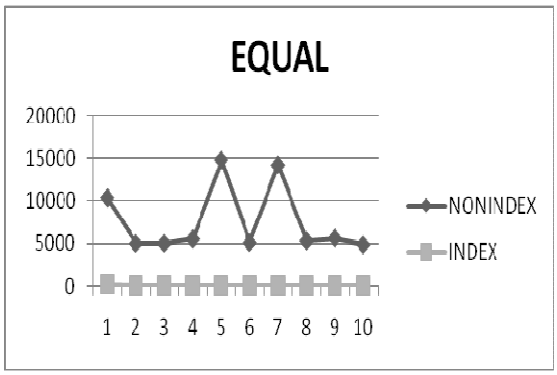

Figure 4 Time Comparison of Query Using Relational Operator

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

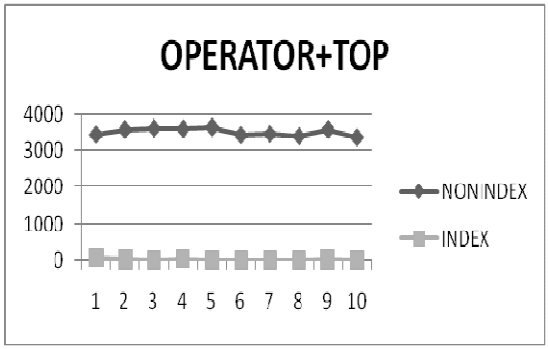

Figure 5 Time Comparison of Query Using Collaboration Operator and TOP

4. CONCLUSION

In Structured Query Languages (SQL), there are many collaboration methods that can be applied to form sub query, such as IN, EXISTS, or Relational Operator sub query. Based on the study, there are the facts have founded:

1. IN and EXISTS has no significant difference on time execution, although IN only return single value and EXISTS return many values from the query result. Likewise indexing is applied in query.
2. Relational operator sub query is not recommended for the query because more time consuming than the other methods of sub query if index are not applied. The situation is inversely if index are applied. Relational operator will extremely efficient than IN and EXISTS methods.
3. Relational operator sub query can be very effective, if the query structured systematically and use different method, i.e. it can be combined with keyword TOP to return query in specific order particularly
4. INDEXING extremely influenced query processing with all methods sub query. The method can be applied to reach out optimal query execution. This can happen because the item in the index was sorted; each indexed item was associated with a physical address. So the process can quickly find a record that user looking for [9]

The above conclusions are based on experiment data. The result may be varied in other environment with different configuration system or volume of data. According to these results, user can specify the effective methods to get an optimal query processing in order to support business process in organization.

REFERENCES

[1] Verma, A. (2011). Enhanced Performance of Database by. IJCSMS International Journal of Computer Science & Management Studies .

[2] Gupta, M. K., & Chandra, P. (2011). An Empirical Evaluation of LIKE Operator in Oracle. *BVICAM's International Journal of Information Technology* .

[3] Connolly, T. M., & Begg, C. E. (2010). Database Systems : A Practical Approach to Design, Implementation, and Management. Boston: Pearson Education.

[4] Mercioiu, N., & Vladucu, V. (2010). Improving SQL Server Performance. *Informatica Economică*

[5] Lungu, I., Mercioiu, N., & Vladucu, V. (n.d.). Optimizing Queries in SQL Server 3008. *Scientific Bulletin – Economic Sciences, Vol. 9 (15)*

[6] Alamery, M., Faraahi, A., Javadi, H. H., Nourossana, S., & Erfani, H. (2012). Application of Bees Algorithm in Multi-Join Query Optimization. *ACSIJ Advances in Computer Science: an International Journal* .

[7] Karthik, P., Reddy, G. T., & Vanan, E. K. (2012). Tuning the SQL Query in order to Reduce Time Consumption. *IJCSI International Journal of Computer Science*

[8] Lasya, S., & Tanuku, S. (2011). A Study of Library Databases by Translating Those SQL Queries Into Relational Algebra and Generating Query Trees. *IJCSI International Journal of Computer Science*

[9] Gillenson, M. L. (2012). Fundamentals of Database Management Systems. USA: Wiley.

[10] Mahajan, D. S., & Jadhav, V. P. (2012). GENERAL FRAMEWORK FOR OPTIMIZATION OF DISTRIBUTED QUERIES. *International Journal of Database Management Systems (IJDMS)*