Enhancing Performance of Database with Improving Query automatically

Maedeh Mansoubi [1], Dr. Amir Masoud Bidgoli [2]

[1] Department of Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran
[2] Department of Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran

Abstract- The present paper deals with improving automatically SQL Server queries in order to increase database performance. This article aims to provide the proposal algorithm called AIQ (Algorithm Improvement of Query) based on program language C# in order to improve automatically the ten queries compiled from various published articles by researchers and also compare the performance of algorithm AIQ with that of SQL Server query optimizer. To achieve these purposes, the mixed method (descriptive and experimental) research design is conducted and then proposal algorithm is provided by researcher and finally query execute time in algorithm AIQ in mode of inactive SQL Server query optimizer is compared with that of in SQL Server in mode of active SQL Server query optimizer on database of Insurance Company. The result obtained shows that speed of improving query with use of proposed algorithm AIQ is more than that of SQL Server query optimizer.

Index Terms- Performance, Database, Query, Automatic

I. Introduction

Nowadays, the use of relational database, in order to save, retrieve and manage in wide range of information, is considered as indispensable prerequisite in our modern world. One of the most powerful and popular databases is SQL Server Software used by most developers and programmers. Application of optimized queries results in quick execution of database operations and enchantment of database performance. In the recent decade, SQL query performance improvement is a very thought-provoking topic between developers and the user community. Users always want a fast response on their data retrieval action and developers put forth their best efforts to provide the method to decrease execution time of query. Many subjects with methods of improving SQL Server query have manually been published in books and different articles so far. Therefore, this article aims to provide a proposal algorithm improving SQL Server queries automatically procedures of proposal algorithm AIQ are as following: in first step, an initial query (not to be improved query) is inserted into program by user as an input data. And next the algorithm analyzes the query; consequently improved query is produced by this algorithm as an output data. It is axiomatic that SQL Server has an optimizer query to select the most optimized execution plan among from plans exist. Therefore, further aim of this paper is to compare performance of algorithm AIQ with that of SQL Server query optimizer. In order to achieve these aims, this paper deals with following sections.

II. Related Work

Stephens, Plew et al (1997) in his article stated that SQL (Structured Query Language) is a standard interactive and programming language for getting information from and updating a database. Although SQL is both an ANSI and an ISO standard, many database products support SQL with proprietary extensions to the standard language. It is also a programming interface.

Bowman (1996) believes that SQL Server queries are used to create, update and retrieve. Specific time is dedicated to any query with regard to syntax and quantity of data in question. According to Fritohey (2012), SQL Server is based on query optimizer and it is important factor for SQL Server to execute complicate queries. The query optimizer creates several execution plans to execute query and then calculate estimate expense for any plan and finally select the best plan for executing that query, in other words; it enhances SQL queries and then produces query with more performance. In the opinion of Schemeling (2010), any database in SQL Server has an item known as auto creates statistics that has two options of "True" and "False". This item (auto create statistic) has a direct relationship with query optimizer. If the option is "True ", query optimizer uses from statistic to select optimization execution plan among of execution plans existed for any query. If the option of item is "False", the query optimizer may not select the most optimized execution plan for queries in question. Schemeling (2010) presented a code about enabling and disabling statistic in SQL Server as following:

Alter database <dbname> set auto_create_statistics [off | on].

In the rest of this section, this paper takes consideration into the studies carried out by different researchers on improving SQL Server queries. Pagie and Bakel (2007) introduced two methods for increasing execution speed of queries, the first is not to use Cartesian for joining to tables and another method is to apply joining of two tables instead of Sub Query. Mercioiu, Vladucu (2010) in the study of Improving SQL Server presented two useful ways to enhance performance of database; first way is to use the operator "Between" instead of operator "In" in conditional expression query. Another is to use operator "Like" instead of operator "Substring" in conditional expression query. Kumari (2012) in his study about improving queries proposed several implications as followings:

1. Applying sysindexes instead of function count (*) to achieve the number of record of table.

2. Applying the operator "Between" instead of operator "Or" in conditional expression query.
3. Using operator "Exists" instead of operator "In" in conditional expression query.

Amjadi Moheb (1392) stated that using conditional expression of "Where" before joining two tables is a useful way to enhance execution speed of query. Mehrotra (2012) also mentioned that applying the in conditional expression query "Not Exists" instead of the in conditional expression query "Not In" may be a useful method to improve query.

III. Methodology

This research is based on descriptive- experimental method. The following procedures are taken for design of study in order to provide the proposal algorithm AIQ and also test it. In this study, two data types are introduced by research. The first one is related to ten queries compiled from different published articles and books in order to be used in proposal algorithm AIQ. The second data used in the study is selected table of Pasargad Insurance Company's real database in order to compare performance of algorithm AIQ with that of SQL Server query optimizer, the compiled queries (first type of data) are accorded to database of company.

A. Data

Specifications of first data type are indicated in following table.

Table 1: list of compiled queries

NO	Improved Query	First Query	Source
1	Select Column1 From Table1 Where Column2=12000000/1 and Column3>1388	Select Column1 From Table1 Where Column2 * 1 = 12000000 and Column3>1388	(Nicolae MERCIOIU, Victor VLADUCU, 2010)
2	Select Column1 From Table1 Where Column3 Between 1391 AND 1393 and Column4=1	Select Column1 From Table1 Where Column3 IN (1391,1392,1393) and Column4=1	(Nicolae MERCIOIU, Victor VLADUCU, 2010)
3	Select Column1 From Table1 Where Column5 Like ' 21%' and Column3=1393	Select Column1 From Table1 Where SUBSTRING (Column5,9,2)=21 and Column3=1393	(Nicolae MERCIOIU, Victor VLADUCU, 2010)-Kumari, 2012)(
4	Select Total_Rows= sum(st.row_count) From sys.dm_db_partition_stats st Where object_name(object_id) = ' Table1' AND (index_id < 2	Select count (*) From Table1	Kumari, 2012)(
5	Select Column1 From Table1 group by Column1,Column2 having Column2 between MAX(Column2) And MIN(Column2)	Select Column1 From Table1 group by Column1, Column2 having Column2>= MAX (Column2) and Column2<= MIN (Column2)	Kumari, 2012)(
6	Select Column2 From Table1 Where exists (Select * From [Table2] Where Column4= Column1) and Column3 =0	Select Column2 From Table1 Where Column1 in(Select [Column1] From [Table2]) and Column3 =0	Kumari, 2012)(
7	Select * From (Select* From Table1 Where Table1.Column1=1)X INNER JOIN (Select * From Table2 Where Table2.PolicyYear=Column2)V on X.Column1=V.Column1	Select Table1.Column1, Table2.Column3 From Table1 INNER JOIN Table2 ON Table1.Column1= Table2.Column1 Where Table1.Column1=1 And Table2.Column4=1386	Amjadi,1392)(
8	Select Table1.Column1 From Table1 INNER JOIN Table2 ON Table1.Column1= Table2.Column1 And Table1.Column2= Table2.Column2	Select Table1.Column1 From Table1,Table2 Where Table1.Column1= Table2.Column1 and Table1.Column2= Table2.Column2	(Bakel,Pagie,2007)
9	Select Table1.Column1, Table1.Column2 From Table1 JOIN (Select Table2.Column1,Table2.Column2, MAX(Table2. Column2) as Col From Table2 group by Table2.Column1, Table2.Column2)X on X.Column1= Table1.Column1 and X.Column2= Table1.Column2	Select Table1.Column1, Table1.Column2 From Table1 Where Table1.Column2=(Select MAX(Table2.Column2) From Table2 Where Table1.Column1= Table2.Column1 and Table1.Column2= Table2.Column2	(Bakel,Pagie,2007)

10	Select Column2 From Table1 Where NOT exists (select * From Table2 Where Column1= Column1) and Column3=0	Select Column2 From Table1 Where Column1 not in(Select Column1 From Table2) and Column3=0	(Mehrotra,2012)

Second data type is table of insurance company's real database whose data storage is OLTP. These tables of database are normalize and organized and name of database called insurance DBU as indicated in following table.

Table2: Specifications of tables of insurance company's database.

NO	Name Table	Number of columns	Number of row	Type column
1	OKCalculation 30001	47	24882704	int-nvarchar(50)-float-
2	OKPolicy	38	1105768	int-nvarchar-float-tinyint-bigint

The tables above have indexes to increase speed of queries execution increasingly.

Table 3: indexes of OKcalculation30001

Column Name	Index Type	Index Name
Age	Non-Unique, Non-Clusterde	IX_AgeCalc
Amount	Non-Unique, Non-Clusterde	IX_AmountCalc
DueDate	Non-Unique, Non-Clusterde	IX_DueDateCalc
PolicyNo	Non-Unique, Non-Clusterde	IX_PolicyNoCalc
PolicyYear	Non-Unique, Non-Clusterde	IX_PolicyYearCalc
Year	Non-Unique, Non-Clusterde	IX_YearCalc

Table 4: indexes of Okpolicy

Column Name	Index Type	Index Name
AgentCode	Non-Unique, Non-Clusterde	IX_AgentCodePol
IssueDate	Non-Unique, Non-Clusterde	IX_IssueDatePol
PolicyEndorNo	Non-Unique, Non-Clusterde	IX_PolicyEndorNoPol
PolicyNo	Clusterde	IX_PolicyNoPol
PolicyYear	Non-Unique, Non-Clusterde	IX_ IX_PolicyYearPol

B. Instruments

This study applied several instruments to achieve aims of the research as in: The researcher used Microsoft Visual Studio.net 2010 to provide algorithm AIQ and program language C# (on windows application) is applied for this purpose. Another instrument used in this study is SQL Server 2008 R2 for queries to be tested and executed. Specifications of personal computer used for the study are as followings:

Personal computer (PC)Intel(R) Pentium (R),CPU G2020 @ 290GHz,2.89 GHz, 1.89 GHz of RAM, System: Microsoft Windows XP Professional, Version 2002, Service Pack 3.

C: Model of Algorithm AIQ

Algorithm AIQ (Algorithm of Improving Query) is designed to improve the compiled queries automatically based on C# program language. This algorithm takes an initial query as an input data and also analyzes syntax of this query. If the query has any syntax error, program signals an error massage; otherwise, the program continues the next step of algorithm. If the query out of the ten compiled query is inserted into the program, the program produces initial query (query not be improved) as an output data for user. If so not, the program analyzes the query and then produces the most improved query as an output data. The flowchart of the algorithm is as in:

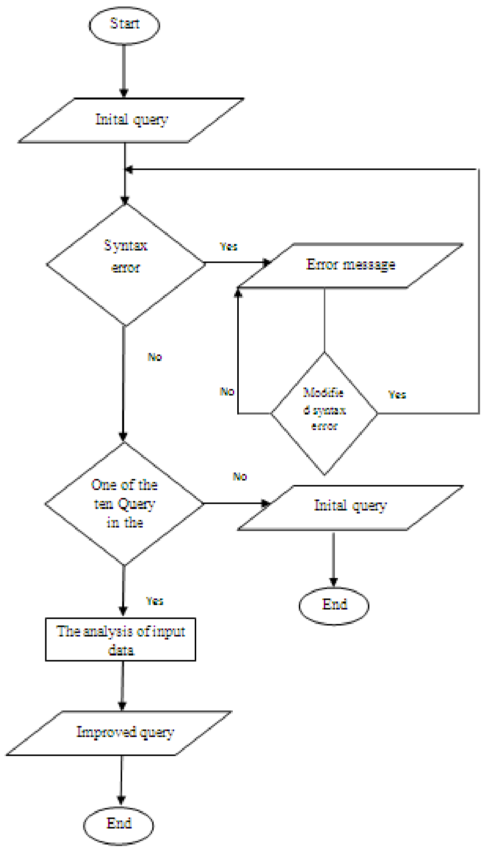

Figure 1: flowchart of algorithm of AIQ

The form of input data for algorithm AIQ is as in:

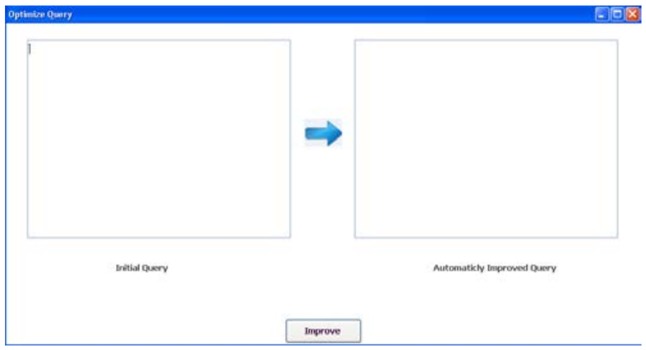

Figure 2: Form of algorithm AIQ input data

After being inserted an initial query in text box and also being clicked on improve button by user, the following functions are executed as in:

```
private void BtnImprove_Click(object sender, EventArgs e)
{
    Stopwatch sw = new Stopwatch();
    sw.Start();
    Improve();
    ExecuteImproveQuery();
    sw.Stop();
    string ExecutionTimeTaken = string.Format
    ("Minutes :{0}\nSeconds :{1}", sw.Elapsed.Minutes, sw.Elapsed.Seconds);
    label3.Visible = true;
    label3.Text = ExecutionTimeTaken;
}
```

In function above, two methods are applied with names of improve () and execute improved query (). The function of improve () is used to improve the initial query and output of this function is considered as an input data for second method. The second function (execute improved query) executes the improved query in the SQL Server Software in option of disable query optimizer (auto-create statistics is off). Finally, the algorithm calculates the time of query execution according to following formula:

Query execution time = Initial query improving time + Improve query execution time in SQL Server

D: Comparison of Algorithm AIQ with SQL Server *Query Optimizer*

In order to achieve second aim; comparing speed of improving initial queries with use algorithm AIQ with that of SQL Server improving initial query with use of query optimizer, the ten compiled queries are modified according to database of insurance company with two different quantities of data (1105768records of table - 24882704 records of table) execution time of the modify query are calculated in three mode as a below: first mode: executing initial query in SQL Server 2008 R2 with mode of active query optimizer.

Second mode: executing initial query in SQL Server 2008 R2 with option of disable query optimizer.

Third mode: execution initial query in algorithm AIQ with option of disable query optimizer.

Table 5: List of modified compiled queries

No	Improved Query	First Query
1	Select PolicyNo From OKCalculation30001 Where Amount = 12000000 / 1 and PolicyYear>1388	Select PolicyNo From OKCalculation30001 Where Amount * 1 = 12000000 and PolicyYear>1388
2	SELECT PolicyNo FROM OKCalculation30001 WHERE PolicyYear Between 1391 AND 1393 and Year=1	SELECT PolicyNo FROM OKCalculation30001 WHERE PolicyYear IN (1391,1392,1393) and Year=1
3	SELECT PolicyNo FROM OKCalculation30001 WHERE DueDate Like ' 21 %' and PolicyYear=1393	SELECT PolicyNo FROM OKCalculation30001 WHERE SUBSTRING(DueDate,9,2)=21 and PolicyYear=1393
4	Select Total_Rows=SUM(st.row_count) FROM sys.dm_db_partition_stats st WHERE object_name(object_id) = 'OKCalculation30001' AND (index_id < 2)	Select count (*) from OKCalculation30001
5	SELECT policyNo FROM OKCalculation30001 group by PolicyNo, Age having Age between MAX(Age) And MIN(Age)	SELECT policyNo FROM OKCalculation30001 group by PolicyNo,Age having Age >= MAX (Age) and Age <= MIN (Age)
6	Select PolicyNo From OKPolicy Where AgentCode = 41011 / 1 and PolicyYear=1387	Select PolicyNo From OKPolicy Where AgentCode * 1 = 41011 and PolicyYear=1387
7	SELECT PolicyNo FROM OKPolicy WHERE PolicyYear Between 1391 AND 1393 and PolicyEndorNo=0	SELECT PolicyNo FROM OKPolicy WHERE PolicyYear IN (1391,1392,1393) and PolicyEndorNo=0
8	SELECT PolicyNo FROM OKPolicy WHERE IssueDate Like ' 21 %' and PolicyEndorNo=0	SELECT PolicyNo FROM OKPolicy WHERE SUBSTRING(IssueDate,9,2)=21 and PolicyEndorNo=0

9	Select Total_Rows= SUM(st.row_count) FROM sys.dm_db_partition_stats st WHERE object_name(object_id) = 'OKPolicy' AND (index_id < 2)	Select count (*) from OKPolicy
10	SELECT policyNo FROM OKPolicy group by PolicyNo, PolicyYear having PolicyYear between MAX(PolicyYear) And MIN(PolicyYear)	SELECT policyNo FROM OKPolicy group by PolicyNo,PolicyYear having PolicyYear >= MAX(PolicyYear) and PolicyYear <= MIN(PolicyYear)
11	select PolicyYear from [OKCalculation30001] where exists (select * from [OKPolicy] where PolicyNo=[PolicyNo]) and PolicyEndorNo=0	select PolicyYear from [OKCalculation30001] where PolicyNo in (select [PolicyNo] from [OKPolicy]) and PolicyEndorNo=0
12	select * from(Select * from OKCalculation30001 WHERE OKCalculation30001.PolicyNo=1)X INNER JOIN (Select * from OKPolicy WHERE OKPolicy.PolicyYear=1386)V on X.PolicyNo=V.PolicyNo	SELECT OKCalculation30001.PolicyNo, OKPolicy.ContractStr FROM OKCalculation30001 INNER JOIN OKPolicy ON OKCalculation30001.PolicyNo=OKPolicy.PolicyNo WHERE OKCalculation30001.PolicyNo=1 And OKPolicy.PolicyYear=1386
13	select OKCalculation30001.PolicyNo from OKCalculation30001 INNER JOIN OKPolicy ON OKCalculation30001.PolicyNo=OKPolicy.PolicyNo And OKCalculation30001.PolicyYear=OKPolicy.PolicyYear	select OKCalculation30001.PolicyNo from OKCalculation30001 , OKPolicy where OKCalculation30001.PolicyNo=OKPolicy.PolicyNo and OKCalculation30001.PolicyYear =OKPolicy.PolicyYear
14	select OKCalculation30001.PolicyNo, OKCalculation30001.PolicyYear from OKCalculation30001 JOIN (Select OKPolicy.PolicyNo, OKPolicy.PolicyYear, MAX(OKPolicy.PolicyYear)as Col from OKPolicy group by OKPolicy.PolicyNo, OKPolicy.PolicyYear)X on X.PolicyNo=OKCalculation30001.PolicyNo and X.PolicyYear=OKCalculation30001.PolicyYear	select OKCalculation30001.PolicyNo, OKCalculation30001.PolicyYear from OKCalculation30001 where OKCalculation30001.PolicyYear = (select MAX(OKPolicy.PolicyYear) from OKPolicy where OKCalculation30001.PolicyNo=OKPolicy.PolicyNo and OKCalculation30001.PolicyYear=OKPolicy.PolicyYear)
15	select PolicyYear from OKCalculation30001 where NOT exists (select * from OKPolicy where PolicyNo=PolicyNo) and PolicyEndorNo=0	select PolicyYear from OKCalculation30001 where PolicyNo not in (select PolicyNo from OKPolicy) and PolicyEndorNo=0

E. Procedure of executing and testing queries in three modes

In order to test queries in the table above in three mentioned mode above, the procedures are carried out respectively as a following:

First mode:

1. Clicking on new query button in SQL Server.
2. Enabling SQL Server query optimizer: Right click on database insurance DBU/properties/option/auto-create statistic= true.
3. Clicking on execute button.
4. Calculating the execute time of initial query.

Second mode:

1. All procedures are carried out like the first mode except in the number 2 of first mode; option of query optimizer must be "False".

Third mode:

1. Enabling SQL Server query optimizer.
2. Debugging proposed program (algorithm AIQ).
3. Inserting initial query in text box of initial query.
4. Clicking on button of improve.
5. The most improved query is indicated on text box of improved query and also execute time of initial query is shown right blue color on from. Finally the researcher describes execute time of query.

IV. Finding and Result

After providing algorithm AIQ by researcher and also executing and testing queries in three modes mentioned in order to compare algorithm AIQ with SQL Server query optimizer, the result obtained are shown:

Table 6: Results of Tested Quires

No	First Mode	Second Mode	Third Mode
1	00:01	00:04	00:00
2	00:08	00:10	00:05
3	00:08	00:12	00:00
4	00:02	00:06	00:00
5	06:29	06:43	06:21
6	00:00	00:01	00:00
7	00:03	00:03	00:00
8	00:00	00:00	00:00
9	00:00	00:00	00:00
10	00:01	00:01	00:00
11	07:20	08:02	06:36
12	00:00	00:00	00:00
13	08:01	10:11	05:33
14	07:42	07:44	06:00
15	06:38	06:49	06:12

The researcher applied the following formula to calculate performance of AIQ and performance of SQL Server query optimizer as in:
Performance = 1/execution time
Execution time of query in first mode = 2193 (Secound)
Execution time of query in third mode = 1847 (Secound)
Performance in first mode = 1/2193 = 4.55
Performance in third mode = 1/1847 = 5.41

The result obtained in this study shows that performance of algorithm AIQ with 0.86 is more than performance of SQL Server query optimizer; furthermore, it presents that some of execution time of queries in algorithm AIQ with 346 seconds is fewer than that of SQL Server query optimizer.

V. Conclusion

The aim of current study was to provide a proposal algorithm AIQ in order to improve the compiled quires automatically and also compare performance of algorithm AIQ with that of SQL Server query optimizer. The conclusion drawn from this study indicated that the speed of improving queries with use of AIQ is more than that of improving query SQL Server query optimizer based on the findings and conclusions illustrated above; the current study also recommends that managers of database, developers, students and programmers can use this proposal Algorithm AIQ to improve queries.

Acknowledgment

I would like to express my sincere gratitude to my advisor, Professor Bidgoli for all the time he has dedicated to this thesis. Last but not the least, I would like to thank to my husband for her love, patience, encouragements, and support throughout these terms took me to accomplish this thesis.

References

[1] Bakel, B. v., & Pagie, R. (2007). Improving Maintenance and Performance of SQL queries. *OCS Consulting* , PaperCC06.

[2] Bowman, J. S. (1996). The practical SQL handbook: using structured query language. *Addison-Wesley Longman Publishing Co., Inc* .

[3] Fritchey, G. (2012). *SQL Server Execution Plans*. Ancestry: Simple Talk Publishing.

[4] Kumari, N. (2012). SQL Server Query Optimization techniques - tips for Writing Efficient and faster Queries. *International Journal of Scientific and Research Publications* , Volume 2, Issue 6.

[5] Mehrotra, S. (2012). SQL Performance Tuning. *globallogic* , 5.

[6] MERCIOIU, N., & VLADUCU, V. (2010). Improving SQL Server Performance. *Informatica Economică* , vol. 14, no. 2.

[7] Schmeling, H. (2010). *Sql server statistics*. Simple-Talk Publishing Cambridge.

[8] Stephens, R. K. (1997). Teach yourself SQL in 21 days, Sams Pub.

Persian References

[1] امجدی محب,ع(1392) افزایش کارایی پایگاه داده با بهینه کردن پرس و جو ها .کارشناسی ارشد , دانشگاه علوم تحقیقات تهران

Authors

First Author – Maedeh Mansoubi, Student of M.Sc. degree in Software Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran,
and m.mansoubi60@gmail.com
Second Author – Amir Massoud Bidgoli is a professor of Computer Engineering in the Department of Computer Engineering, Islamic Azad University, and Tehran North Branch. He received his B.Sc. degree in Electronics Engineering from Lancaster University (UK) and the M.Sc. degree in Computer & Control Engineering from Salford University (UK), in 1987 and 1989, respectively. He received his Ph.D. in Computer Science from Salford & Manchester University (UK), in 1996. He is a member of IEEE and is indexed in IEEE Explorer. His research interests include computer networks & evolutionary algorithms, computer architecture, networking, distributed systems and image processing. He has over fifty publications in journals and ISI conferences and drbidgoli@gmail.com