International Journal of Database Theory and Application
Vol.8, No.6 (2015), pp.133-138
http://dx.doi.org/10.14257/ijdta.2015.8.6.12

Efficient Algorithm for Multi Query Optimization

L. J. Muhammad[1], Yahaya Bala Zakariyau[2] and Abdullahi Garba Ali[3]

[1,2]*Mathematics & Computer Science Department, Federal University, Kashere*
Computer Science Department, Bayero University, Kano[3]
lawan.jibril@fukashere.edu.ng[1], *baladada57@yahoo.com* [2] *jgewel@yahoo.com* [3]

Abstract

Multi Query Optimization is an important process in database and it becomes the commonplace due to the frequent usage of decision support systems in almost all the multinational enterprises. The multiple queries from different users that have been addressed to one schema often have a lot of common sub-expressions and it is the function of the multi query optimization algorithms such as Basic Volcano, Volcano RU and Volcano SH algorithms to optimize such multiple queries together and executes the common operation once and share the output among the queries. In this work, a multi query shareability algorithm which can efficiently detect the common sub-expressions among the multiple queries and share the output among those queries was proposed and algorithm for optimal order of those queries was also proposed. The Algorithm has a time complexity of $O(n^2 + 9n +6)$ while the most recent basic algorithm thus Volcano RU Algorithm has $O(2n^2 +20n +12)$, both the algorithms have $O(n^2)$ time complexity which is quadratic in nature. However, the Proposed Algorithm is more efficient and better than Volcano RU algorithm even if n approach to infinity.

Keywords: *Algorithm, Multi-Query, Optimization, inter-query shareability, common sub-expressions, Time Complexity, Matrix*

1. Introduction

In Query optimization phase, the best execution plan is generated, this is done by putting into account the resources required to execute the query as well as the resources required to get the plan and database statistics are used to make appropriate decisions. But In multi-query optimization, are queries are not optimized one by one; instead, they are optimized and hence executed in batches [5].

2. Multi-Query Optimization (MQO)

In multi query optimization, queries are optimized and executed in batches. Individual queries are transformed into relational algebra expressions and are represented as graphs According to [7] the graphs are created in such a way that:

i. Common sub-expressions can be detected and unified;
ii. Related sub-expressions are identified so that the more encompassing sub-expression is executed and the other sub-expressions are derived from it. For example if We have $\sigma A \leq 5(E)$ and $\sigma A \leq 10(E)$, $\sigma A \leq 10(E)$ is executed and $\sigma A \leq 5(E)$ derived from it.

The optimizer therefore concentrates on:-

i. Identifying the common expressions so that the database accesses and computational costs are minimized;
ii. Identifying groups of sub-expressions where internal derivations are possible so that derivations can be made and database access is minimized;
iii. Get a search strategy so that the search is not too long.

ISSN: 2005-4270 IJDTA

3. Multi-Query Optimization Algorithms

Multi-query optimization algorithms are optimizing queries not one by one but instead in batches. There are three basic multi query optimization algorithms which are the bedrock of all the other multi-query optimization algorithms, these algorithms are Basic Volcano, Volcano SH, and Volcano RU algorithms. They use DAG to represent the search space. In some cases however, search space is represented as an AND-OR DAG [11].

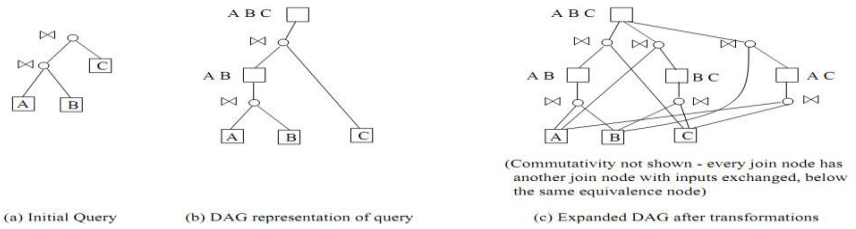

Figure 1. Query Representation: Tree, DAG and Extended DAG

An AND-OR DAG is a DAG whose nodes are divided into two: the AND nodes and the OR nodes. AND nodes have only OR nodes as their children and OR nodes have only AND nodes as their children. The AND node in an AND-OR DAG has algebraic operation like select (σ), project (π), *etc.* They are therefore referred to as operational nodes. The OR node of an AND-OR DAG represents a logical expressions that generates the same result set as when a child operational node is applied on its children/child. OR nodes are referred to as equivalence nodes. The expanded DAG is used as a representation for modern optimizers because they are easily extensible. [5]

3.1. Basic Volcano

The Basic Volcano Algorithm was proposed by [6] and it uses DAG as a representation of the query plans. It has a problem of extensibility since AND-OR DAGs are easier to extend than the DAGs. [5]The Basic Volcano algorithm materializes all nodes that appear more than once. This brings in a problem that not all nodes that appear more than once cause savings when materialized. As observed in [5] for some nodes, it is cheaper to re-compute than to materialize and reuse them. This is because materialization involves writing and reading to disk which is costly.

3.2. Volcano SH

Volcano-SH is an extension of the Basic Volcano algorithm in that it uses the Basic Volcano optimal plans as an input. The volcano SH computes the cost of each node and decides whether or not it is cost effective to materialize it. This is done by considering a scenario of materialization and reuse against re- computation. [7]

3.3. Volcano RU

The Volcano-RU exploits sharing well beyond the optimal plans of the individual queries. Though volcano SH algorithm considers sharing, it does it on only individually optimal plans therefore some sharable components which are in sub-optimal plans are left out. Including sub-optimal states however implies that the sample space of the nodes has to increase. The search algorithm must be able to put it into consideration so that the searching cost is still below the extra savings made. The volcano RU algorithm aims at

reusing and sharing sub-expressions which are not necessarily in the individual query optimal plans [1].

4. Multi Query Shareability Problem

While searching for the sharable sub-expressions between queries, we are guided by the three cases in which the sub-expressions can be reused [2].

i. If the sub-expressions are exactly similar, then we can be able to use any of the sub-expressions in all the instances where it exists.
ii. If however the case is not so, and one of the sub- expressions is a super set of the rest, We then need to make a decision as to what sub-expression should be actually executed and what sub- expression is to be derived from the other.
iii. In case where the sub-expressions are not subsets of equivalent nodes, so we have to look for another sub-expression to be created so that the rest are to derived from it. However, for the interest of identification, once pair of queries has any such sub-expressions, we say they are sharable.

4. Efficient Multi Query Optimization Algorithm

4.1. Multi Query Shareability Algorithm

A matrix was used to design new efficient multi query shareability algorithm, In this algorithm, a comparison of two query plans at the same time for each pair of nodes was done, so as establish whether or not they are sharable. If they are sharable, the appropriate entry in the multi query sharing matrix is incremented. Let us consider a situation where we have a batch of n queries Q_1, Q_2,…… Q_n. For any query Q_x, a set of nodes in the Volcano optimal plan can be got by the method proposed by [6] The input, like in the Volcano-SH is a set of volcano best plans. Let S_x be the set of equivalence nodes in the plan of Q_x. The algorithm checks nodes pairwise and establishes whether sharing can be possible. If it is possible for any queries Q_x and Q_y , the query popularities and M[x,y] are incremented. This is done until all sets are exhausted. The algorithm traverses the pseudo-rooted DAG and gives a summary of the extent of sharing between any two queries. Using this information, the popularity can be deduced. The matrix acts as a guide to establish which queries have sub-expressions in common and which queries are disjoint. Considering a general query sharing matrix M below:

$$\begin{Bmatrix} M1_1 & M1_2 \dots\dots\dots & M1_n \\ M2_1 & M2_2 \dots\dots\dots & M2_n \\ Mn_n & Mn_n \dots\dots\dots & Mn_n \end{Bmatrix}$$

The entry M [x, y]= mxy is an integer that shows how many sub-expressions (equivalence nodes in AND-OR DAGs) that are sharable between queries Q_x and Q_y . This is the same as the value of M [y, x]. The sum up of the entries in a column (or a row), will result the total number of instances in which nodes in the plan have sharable partners. This is called query popularity. All nodes that have partners are marked so that the optimizer can identify which nodes necessitate checking other plans while searching for common sub-expressions and which nodes do not. This helps in eliminating null searches hence a more efficient strategy. After identifying the extent of sharing among the queries, we can be able to tell which pair or pairs of queries have nothing in common. For such queries, we do not need to search for common sub- expressions since they do not exist. Likewise, we can be able to tell for each query, which other queries in the batch share nodes with it and to what extent. When start searching for common sub-expressions, it will start with one query and search for the sharable sub-expressions with other queries in the DAG. The Query plan, which is at the center of the searching process, is called the focal node. Since the plans are already traversed at the searching stage, the nodes will be

marked (those with sharable sub-expressions elsewhere) and those node that will not be marked. It is only on marked nodes that the search for other plans for sharable nodes will be done. It should be noted that a node being marked does not mean that its partners will be searched for in all plans. Since the query sharing matrix has the summary of what query shares with what, only those with non-zero entries in the sharing matrix are searched for a column/ row representing the extent of sharing between the focal plan and such a plan are checked.

4.2. Efficient Optimal Order Algorithm

This Proposed algorithm unlike the Volcano-RU is to use the Basic Volcano optimal plans as the inputs; it uses order in another way. It use identify the plan with the highest popularity and establishing its details (node costs and frequency). The common sub-expressions are searched for in the rest of the queries following the stable marriage preference list of the focal plan. If according to the sharing matrix M, a certain plan is not having any node to share with the focal plan, we do not search it. However to reducing the sample space without affecting the output and optimal plan will minimize the optimization cost in the algorithm. Identifying more than one node to be sharable, this will enable to decide which of the nodes will remain in the plan and which (the rest) is/ are to be removed from the plan. Action is done such that then plans are supplied with the output of the retained node so that the output of the removed nodes is compensated. In an AND-OR DAG, the equivalence node will have multiple parents. This reduces the sample space for the subsequent searches without affecting the output of the query batch. It also saves the cost of re-computation.

When a focal plan is identified, the costs and number of times a node is used in the plan, count are established. If a node is marked, then it means that it has sharable partners in other plans. The sharable nodes are searched for in the plans that share at least a node with the focal plan. Each time a sharable node is found, the count is incremented by one and it's assured to be of the same cost as the node in the focal plans. The test for materialization used in the Volcano SH is then applied using the new estimated cost and the node count in the pseudo rooted plan. In this algorithm, nodes were chosen to materialize from the high order nodes downwards. This is used in such a way that if say two nodes are sharable and two have to be removed, this reduces the sample space of subsequent searches and saves the cost of computing already catered for nodes.

5. Evaluation Analysis of the Algorithms

Time complexity is a scientific framework to measure the amount of times required to execute an algorithm. It also expresses the relationship between the size of the input and the run time for the algorithm. However, the time complexity of the Proposed and Volcano RU Algorithms were determined so as to identify the best algorithm among them. Both algorithms were linear, where the Proposed Algorithm has $O(n^2+ 9n+6)$ time complexity whereas Volcano RU Algorithm has $O(2n^2+20n+12)$ time complexity. Table 1.1 shows the time complexity of both algorithms considering the n size and Chart 1.1 depicts the graphical representation of the result.

Table 1.1. Time Complexity of the Algorithms

Input n size	Volcano RU (ms)	Proposed Improved Optimization Algorithm (ms)
1	34	16
10	412	196
100	220011	10906
1000	2020012	1009006

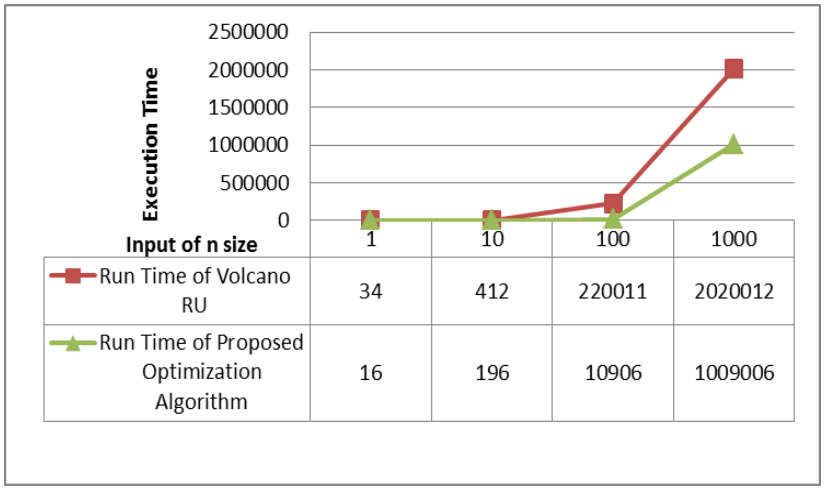

Chart 1.1. Evaluation Analysis Result

6. Conclusion and Future Work

The Proposed Algorithm addresses the weaknesses identified in the Basic Volcano, Volcano SH and Volcano RU. It is against these addressed weaknesses, coupled with the strengths already exhibited by these algorithms that a lay a foundation the new algorithm and that what made Algorithm being better than Volcano RU algorithm in term of the complexity of time even if the n approach to infinity. The correctness of the proposed improved algorithm was proved because it halts and its precondition (the set of plans that make up the virtual DAG in a decreasing order of their popularity) combined with the conditions arising from execution lead it is post condition (Optimized Plans). Therefore, the objective function of the algorithm was achieved thus cost saving in terms time complexity was achieved. However, the evaluation done on the proposed algorithms against the basic multi query optimization algorithms is out coding. Therefore in future work, there is need to code the proposed algorithms and run them with the existing ones (such as Basic Volcano, Volcano-RU, Volcano-SH) on similar data and similar hardware. This will give the runtime comparison with the existing schemes.

References

[1] Dalvi J. R., Raghavan, V. V. and Brian B., "Analysis of Common Sub-expression Exploitation Models in Multiple Query Processing", in Proc. 10th Int. Conf. on Data Engineering, IEEE Press, **(2001)**, pp. 82-97.
[2] Graefe G. and McKenna W. J., "Extensibility and Search efficiency in the Volcano Optimizer generator", Technical report CU-CS-91-563. University of Colorado., **(1991)**.
[3] John N., "On Query Optimization in Relational Databases", A Dissertation Submitted to Makerere University, Uganda, **(2004)**.
[4] Kyuseok S., Sellis T. K and Nau D., "Improvements on a Heuristic algorithm for Multiple-query Optimization", Technical report, University of Maryland, Department of Computer Science, **(1994)**.
[5] Muhammad L. J., Y. B. Zakariyau, A. G. Ali and A. Garba, "On Multi-Query Optimization Algorithms Problem", International Journal of Database Theory and Application, Science & Engineering Research ISSN: 2005-4270, vol. 7, no. 6, **(2014)**, pp.13-20.
[6] Park P. and Segar A., "Using common sub-expressions to optimize multiple queries", Proceedings of the IEEE International Conference on Data Engineering, **(1988)**.
[7] Roy P., Seshadri S., Sudarshan S. and Bhobe S., "Practical Algorithms for multi query Optimization", Technical report, Indian institute of Technology, Bombay, **(1998)**.
[8] Pandao S. D. and Isalkar A. D., "Multi Query Optimization Using Heuristic Approach", International Journal of Computer Science and Network (IJCSN), vol. 1, no. 4, **(2012)**.