ISSN No. 0976-5697

Volume 3, No. 3, May-June 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Study on Optimization Techniques and Query Execution Operators That Enhances Query Performance

Tejy Johnson*
Research Scholar, Department of Computer Applications,
Dr.MGR University, Maduravoyal
Chennai, India
Kty07_1980@yahoo.co.in

Dr.S.K.Srivatsa
Senior Professor, Department of Computer Applications
St. Joseph College of Engineering, Chennai-119
Chennai, India
profsks@rediffmail.com

Abstract: Now a day's Query Optimization is a key technology for every application from operational systems to data warehouse and from analytical systems to content management systems. Query Optimization is of great importance for the performance of a relational database, especially for the execution of complex SQL statements. The performance of database systems is critically dependent upon the efficiency of Optimization techniques and Query Execution Operators. Thus we present a detailed study of the various Optimization techniques and Query execution Operators that helps to enhance the query performance. A survey of the available work into this field is also given. This detailed study helps to choose an Optimization strategy that best suits your query environment. This study facilitates the enhancement of query performance by determining the best optimization strategy and helping us to build a query optimizer model. In this paper we focus on the various Optimization techniques, Components of query optimizer and the Query execution operators that are used for the enhancement of query performance.

Keywords: Query Optimization, Query Optimizer, Query Optimization Techniques, Query Execution Operators and Optimization Strategy

I. INTRODAUCTION

Now days we have to deal with a increasing amount of facts, figures and data. Therefore it is necessary to store this information in an adequate way. Thus the significance of database system is increasing day by day. To meet the current trend we need an efficient Query Optimizer that would help us to determine the best Optimization Strategy. The development of query optimization technology plays a vital role in the success of commercial database systems. In general Query Optimization [1] refers to the process of producing an optimal execution plan for a given query where optimality is with respect to a cost function to be minimized. The cost of optimizing a query is mainly incurred by the investigation of the solution space for alternative execution plans. As an optimizer faces different query types with different requirements, it should be easy to adapt the search strategy to the problem, which implies some form of extensibility. A desirable Optimizer [2] is one where the search space includes plans that have low cost, the costing technique is accurate and the enumeration algorithm is efficient. The Search Space [3] for optimization depends on the set of algebraic transformations that preserve equivalence and the set of physical operators supported in an optimizer. These transformations do not necessarily reduce cost and therefore they must be applied in a cost based manner to ensure a positive benefit. The optimizer may use several representation of a query during the life cycle of optimizing a query. The initial representation is often the parse tree of the query and the final representation is an operator tree.

In this paper we will discuss some of the ways in which queries can be optimized. Here we mainly focus on the actual objective of the Optimizer that is to find a Strategy close to the optimal. In section 2 we discuss about the Search Space, Search Strategy and Processing Tree which describes the Query Optimization problem and the associated cost model. In section 3 we discuss about the various Optimization Techniques that are used for query Optimization. In section 4 we discuss about the various Query Execution Operators that enhances the query performance. Finally in section 5 we conclude by suggesting a Query Optimizer model which is best suited to enhance the performance of the query.

II. GENERAL PRINCIPLES OF QUERY OPTIMIZATION

Query Optimization [4,16] can be formally defined as a process of transforming a query into an equivalent form which can be evaluated more efficiently. The goal of Query Optimization is to find an Execution Strategy for the query that is close to the optimal. An Execution Strategy for a distributed query can be described with relational algebra operations and communication primitives. The Query Optimizer [5] that follows this approach consists of three main components: - a Search Space, a Search Strategy and a Cost model. The Search Space is the set of alternative execution to represent the input query. In other words the Search Space or Solution Space is the set of all Query Evaluation Plan's that compute the same result. A point in the solution space is one particular plan i.e., solution for the problem. A solution is described by the Query tree for executing the join expression.

Every point of the search space has a cost associated with it, a cost function maps query trees to their respective cost. The Query tree itself is a binary tree that consists of base relations as its leaves and joins operations as its inner nodes. Edges denote the flow of the data that takes place from the leaves of the tree to the root. The specification of the

optimization search space is influenced by the input query and the nature of investigated Query tree.

The Search Strategy [6] explores the search space and selects the best plan. It defines which plans are examined and in which order. The cost model predicts the cost of a given execution plan which may consist of the following components:-

a. ***Secondary Storage Cost: -*** This is the cost of searching data on the secondary storage.

b. ***Memory Storage Cost: -***This is the cost pertaining to the number of memory buffers needed during Query execution.

c. ***Computation Cost: -*** This is the cost of performing memory operations on the data buffers during Query Optimization.

d. ***Communication Cost: -*** This is the cost of shipping the query and its results from the database site to the site where the query originated.

Finally Cost functions provide the basis for comparing different Query Evaluation Plans and for choosing the best plan for execution. Cost functions reflect various aspects of the execution environment such as distribution, CPU consumption, sizes of tables, I/O costs etc. The cost of optimizing a query is mainly incurred by the investigation of the solution space for alternative execution plans. As the solution space gets larger for complex queries, the search strategy that investigates alternative solution is critical for the Optimization cost.

Traditional Query Optimization [7,17] uses an Enumerative Search Strategy which considers most of the points in the solution space. Enumerative strategies can lead to the best possible solution, but face a combinatorial explosion for complex queries. In order to investigate larger spaces Randomized Search Strategies have been proposed. Randomized Search Strategy concentrate on searching a local optimal solution around some particular points. They consist of two steps: - First, one or several start solutions are obtained by Depth first search. Second, the start solutions are improved until local optimal solutions are obtained. In this phase neighboring solutions are randomly obtained by applying transformations. A local optimal solution is the one which is the best among all the neighboring solutions. Randomized strategies involve the definition of several parameters (the number of transformations to apply and the criterion for considering a solution to be a local optima one, etc).

The other approach is the Genetic Strategies [8] that start with a population of solutions from which new generations are built by successively applying Crossovers to individuals of the original population. The basic actions in Genetic Strategies are Crossover and Mutation. A Crossover consists in selecting two parents from a population of complete processing trees and generating two off springs according to some principle. The individuals to be crossed are chosen at random, but the choice is biased by their fitness. The fitness is related to the function to be optimized. Thus new generation are expected to contain better individuals than the previous ones, because they are built from the features of the selected parents. A Mutation applies to a unique individual and has the same nature of transformations in Randomized Strategies. The incremental nature of processing tree nodes is also important for efficiently implementing Crossover actions.

Most Search Strategies [9] fall into two main classes: Exhaustive Strategies and Heuristic Strategies. Exhaustive Strategies generate all plans possible by the creation procedure. Most commonly known are Depth first search strategy and Breadth first search strategy. The Heuristic is the most commonly known domain independent that limits the creation of Query Evaluation plan in each level to one choice. There are three major aspects common to all Search Strategies. First, any Search Strategy needs to determine when the search should terminate. Second, the Search Strategy has to decide where to continue the search when several choices are given. Finally, the Search Strategy might discard certain choices thus not considering them for further exploration.

III. AN OVERVIEW OF QUERY OPTIMIZATION TECHNIQUES

Query Optimization [10, 11, 18] is defined as the activity of choosing an efficient strategy for processing a query. The main aims of Query Optimization are to choose a transformation that minimizes resource usages, reduce total execution time of query and also reduce response time of the query. There are three important components of query optimization that are Access method, Join criteria and Transmission cost.

a. ***Access Method:*** In most database system tables can be accessed in one of two ways: by completely scanning the entire table or by using an index. The best access method to use, always depend upon the circumstances.

b. ***Join Criteria:*** If more than one table is accessed the manner in which they are to be joined together must be determined.

c. ***Transmission Cost:*** If data from multiple sites must be joined to satisfy a single query then the cost of transmitting the results from intermediate steps needs to be factored into the equation.

Query Optimization [12, 13] is the process of selecting the most efficient Query Evaluation plan for a query. There are two main techniques for Query Optimization. The first approach is to use a Rule based or Heuristic based method for ordering the operations in a Query Execution Strategy. The rules usually state general characteristics for data access. The second approach systematically estimates the cost of different Execution Strategies and chooses the least cost solution. This approach uses simple statistics about the data structure size and organization as arguments to a cost estimating equation. In practices most commercial database systems use a combination of both techniques. Hence Optimization is the process of choosing the most efficient way to execute a SQL statement.

The various Optimization techniques [14, 15, 19] used to obtain efficient execution plan are as follows: Heuristic Optimization, Syntactical Optimization, Cost based Optimization and Semantic Optimization. Heuristic Optimization: - It is a rule based method of producing an efficient query execution plan. Because the query output of the standardization process is represented as a canonical query

tree, each node of the tree maps directly to a relational algebraic expression. The function of heuristic query optimizer is to apply relational algebraic rules of equivalence to this expression tree and transform it into a more efficient representation. Using relational algebraic equivalence rules ensure that no necessary information is lost during the transformation of the tree.

The major steps [20, 21] involved in Heuristic optimization are:

Step 1: Break conjunctive selects into cascading selects.

Step 2: Move selects down the query tree to reduce the number of returned tuples.

Step 3: Move projects down the query tree to eliminate the return of unnecessary attributes.

Step 4: Combine any Cartesian product operation followed by a select operation into a single join operation.

When these steps have been accomplished the efficiency of a query can be further improved by rearranging the remaining select and join operations so that they are accomplished with the least amount of system overhead.

a. Syntactical Optimization:- It relies on the users understanding of both the underlying database schema and the distribution of the data stored within the tables. All tables are joined in the original order specified by the user query. The Optimizer attempts to improve the efficiency of these joins by identifying indexes that are useful for data retrieval. This type of Optimization can be extremely efficient when accessing data in a relatively static environment. Using Syntactical Optimization indexes can be created and tuned to improve the efficiency of a fixed set of queries. Problems occur with Syntactical Optimization whenever the underlying data is fairly dynamic.

b. Cost based Optimization:- To perform Cost based Optimization an Optimizer needs specific information about the stored data. This information is extremely system dependent and can include information such as file size, file structure types, available primary and secondary indexes and attributes selectivity. The goal of any Optimization process is to retrieve the required information as efficiently as possible. The realistic goal of a Cost based Optimizer is not to produce the optimal execution plan for retrieving the required data, but is to provide a reasonable execution plan.

c. Semantic Optimization: - It operates on the premise that the Optimizer has a basic understanding of the actual database schema. When a query is submitted the Optimizer uses its knowledge of system constraints to simplify or to ignore a particular query if it is guaranteed to return an empty result set. This technique holds great promise for providing even more improvements to query processing efficiency in future relational database systems.

IV. QUERY EXECUTION OPERATORS

In this section we see how the Operators help in the Query Optimization [22, 23]. The various operators discussed are as follows:

a. Selection operation: - The lowest level Query processing Operator for accessing data is the file scan. It searches and retrieves record for a given selection condition. The two scan algorithms used to implement the selection operation are linear search and binary search. In a linear search, the systems scans each file block and test all records to see whether they satisfy the selection condition. If the file is ordered on an attribute, and the selection condition is an equality comparison on the attribute, then we can use a binary search to locate records that satisfy the selection. Selection algorithms that use an index are referred as index scans. Index structures also referred as access paths, since they provide a path through which data can be located and accessed. Selections specifying an equality condition can use a secondary index. Selections that specify equality comparison on a key attribute can use primary index.

b. Sorting: - It plays an important role in database systems for two reasons. First, SQL queries can specify that the output be sorted. Second, several of the relational operations such as joins can be implemented efficiently. We can sort a relation by building an index on the sort key and then using that index to read the relation in sorted order. Sorting of relations that do not fit in memory is called external sorting. The most commonly used technique for external sorting is external sort merge algorithm.

c. Join operation: - Join operation is the most important one in Query Optimization. It refers to the process of calculating the optimal join order that is the order in which the necessary tables are joined, when executing a query. A join combines records from two tables based on some common information. The order of join is a key factor in controlling the amount of data flowing between each operator in the execution plan. The order in which the tables are joined determines the cost and performance of the query. The nested loop join requires no indices and it can be used regardless of what the join condition is. In a nested loop join if an index is available on the inner loops join attribute, index lookups can replace file scans. This join method is called indexed nested loop join. Merge joins can also be used to compute natural joins and equi-joins. Like Merge join, Hash join can also be used to implement natural joins and equi-joins. In hash join, a hash function is used to partition tuples of both relations.

d. Projection operation: - The projection operation is used to select data of particular attributes (columns) from a single relation and discards the other columns. We can implement projection easily by performing projection on each tuple, which gives a relation that could have duplicate records, and then removing duplicate records.

e. Set Operation: - We can implement the union, intersection and set difference operations by first sorting both relations and then scanning once through each of the sorted relations to produce the result. For all these operations, only one scan of the two input relations is required.

f. ***Aggregation Operation: -*** We can implement aggregate operations sum, min, max, count and avg on the fly as the groups are being constructed. The cost estimate for implementing the aggregate operation is the same as the cost for aggregate functions such as min, max, sum, count and avg.

The query performance can be measured by using three main metrics: - Query cost, Page read and Query execution time.

g. ***Query Cost: -*** which is a measure of the CPU and the I/O resources used by the query.

h. ***Page Read: -*** They are a set of statements that indicates how many pages were read from storage.

Query execution time: - It measures the total execution time of a query in milliseconds.

The other ways to improve the query performance are rewriting the query, normalizing or de normalizing tables and adding indexes. Joins in query increase the cost of the query and thereby loses its performance. Hence it is best to minimize the number of join clauses to ensure optimized query performance. Outer joins incur more cost than inner joins. So avoid using outer joins. Based on this study we propose query optimizer model which is discussed in coming section.

V. PROPOSED QUERY OPTIMIZER MODEL

The Query Optimizer [26-30] determines the most efficient way to execute a SQL statement after considering many factors related to the object referenced and the conditions specified in the query. This determination is an important step in the processing of any SQL statement and can greatly affect the execution time. By default the goal of the Query Optimizer is the best throughput.

To maintain the effectiveness of query optimizer you must have statistics that are representative of the data. The Query optimizer operations include transforming queries, estimating and generating plans. The components of query optimizer are depicted in the following figure 1.

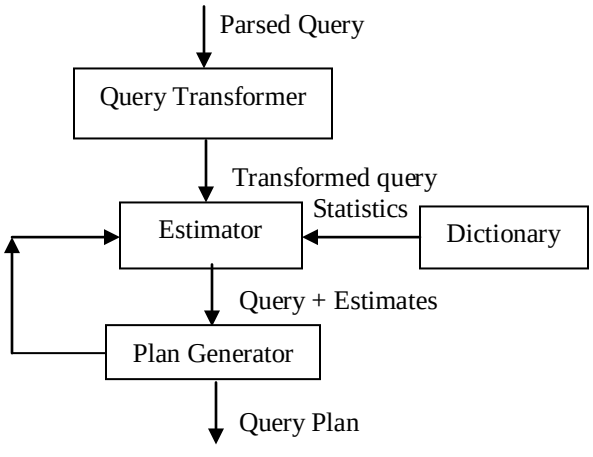

Figure. 1 Components of Query Optimizer

The input to the query transformer is a parsed query which is represented by set of query blocks. The query blocks are nested or interrelated to each other. The main objective of the query transformer is to determine if it is advantageous to change the form of the query so that it enables generation of a better query plan. The end goal of the estimator is to estimate the overall cost of a given plan. The estimator generates three different types of measures that are selectivity, cardinality and cost. Selectivity represents a fraction of rows from a row set. Cardinality represents the number of rows in a rows set.

Whereas Cost represents units of work or resource used. The query optimizer uses disk I/O, CPU usage and memory usage as unit of work. These measures are related to each other and one is desired from another. If statistics are available then the estimator uses them to compute the measures. The statistics improve the degree of accuracy of the measures. The main function of the plan generator is to try out different possible plans for a given query and pick the one that has the lowest cost. Modern database systems use a query optimizer to identify the most efficient strategy called plan to execute declarative SQL queries.The query optimizer performs the following steps:-

a. The optimizer generates a set of potential plans for the SQL statement based on available access paths and hints.
b. The Optimizer estimates the cost of each plan based on statistics in the data dictionary for the data distribution.
c. The Optimizer calculates the cost of access path and joins orders based on the estimated computer resources.
d. The Optimizer compares the costs of the plans and chooses the one with the lowest cost.

After the analysis of different Optimization techniques, execution operations according to the three fundamental aspects of creation, search strategy and cost function, we show here how we can combine them for the implementation of an Optimizer component. To keep our explanations simple and clear, we prefer describe the model informally. This model is independent of the approach taken to optimize the query. This model can be used as a general architecture for implementing an extensible query optimizer.

This optimization model consists of following major components: the Creator, the Cost evaluator, the Updater, the Tester and the Decider. The Creator component takes as its input the search tree and selects a state that is to be expanded next. Its output is a set of new states that form the input to the component cost evaluator. This component determines s the cost of each new state. Then the component updater decides which state to be eliminated or /and which state to be appended from the newly created search tree. This decision can be based on comparing the costs of different query evaluation plan created so far. Since the search tree has changed the Tester might decide to either terminate the search or output one query evaluation plan or to continue the creation process. In case the search continues, it is the responsibility of the decider to determine the next state to be expanded. The components Tester, Decider and Updater only depend on the representation of the search tree and they do not need to know the detailed representation of a state itself.

The only relevant information needed from a state are the cost values of generated query evaluation plans. On the other hand the component Creator depends only on the representation of the states. This separation and independence is advantageous in case a new search strategy should be

implemented, cost function should be changed or rewriting rules are changed, deleted or added. All of these changes can be performed in the corresponding component without affecting others. The Creator either generates different query evaluation plan for accessing one relation considering the different access paths available or adds to an existing query evaluation plan another relation by creating a join operator and considering different join methods such as nested loop join, an index join or a sort merge join. It uses an internal cutoff to reduce the number of plans it tries when finding the one with the lowest cost. The cutoff is based on the cost of the current best plan. If the current best cost is large then the Creator tries harder to find a better plan with lower cost. If the current best cost is small then the creator ends the search swiftly. The Cost evaluator then determines the cost of each newly created query evaluation plan using the cost functions. The Updater depending on properties of already existing or newly created plan, it stores the cheapest plan and all other plans are eliminated. The Decider checks if all paths in the search tree have been explored. If so the final plan is the output, otherwise the search continues.

We believe that our model provides a well founded basis for implementing an extensible Optimizer. One can change the search strategy, the cost functions or the query evaluation plan creation depending on new or changing requirements without affecting the components of the Optimizer.

VI. CONCLUSION AND FUTURE WORK

In this paper we have discussed about various Query Optimization techniques, various Query Execution Operators and Search Strategies used for optimizing a query. We have also discussed about the basic principles of Query Optimization. We have tried to quantify the factors that enhance the performance of the query. We have also proposed a simple operational model for Query Optimization that incorporates the modularity and flexibility necessary to implement an extensible Query Optimizer as required for the new generation of database management systems. We believe that results of this paper will help to extend the existing Optimization techniques to the demanding requirements of new application areas. Usage of constraints and methods like introducing an additional join to the original query requires further study from an implementation point of view.

VII. REFERENCES

[1] G.M. Sacco and S.B. Yao, "Query Optimization in Distributed Database Systems," In Advances in Computers, Academin Press, New York, vol. 21, pp. 225–273, 1982.

[2] L. Mackert and G. Lohman, "R* Optimizer and performance evaluation for distributed queries," In Proceedings VLDB , Kyoto, Japan , Aug 1986, pp.149–159.

[3] A. Aljanaby, "A Survey of Distributed Query Optimization," The International Arab journal of Information technology,vol. 2, Jan 2005, pp.48–57.

[4] M. Jarke and J. Koch, "Query Optimization in Database systems," aCM Computing surveys, June 1984,pp. 111-152.

[5] S.G. Rosana ,Lanzelotte,Patrick Valduriez," Extanding the Search Strategy in a Query Optimizer," In proceedings of 17th VLDB, 1991,pp. 363-373.

[6] A. Makinouchi,M. Tezuka,H. Kitakami and S. Adachi, "The Optimization Strategy for Query Evaluation in RDB," In proceedings of 7th International conference on VLDB,IEEE ,New York, vol. 1, pp. 518–529, Sept 1981.

[7] A.M.J. Skulimowski,"Optimal Strategies for Quantitative data retrieval in Distributed database systems," IEEE 2nd International conference on Intelligent systems Engineering, 1994, pp. 389-394.

[8] W. Xu and U.M. Diwekar," Improved Genetic algorithms for deterministic Optimization and Optimization under certaintity," Industrial and Engineering chemistry research, ACS publications, Aug 2005,pp.7138-7146.

[9] A. Hameurlain and Franck Morvan, "Evolution of Query Optimization methods," Transactions on large scale data and knowledge centred systems, vol. 5740/2009, Sep. 2009,pp.211-242.

[10] Y.E. Loannidis, "Query Optimization," The computer science and Engineering handbook,CRC press, pp. 1038-1054,1996.

[11] D. Kossmann, "The State of art in Distributed Query Optimization," ACM computing surveys, Sep 2000.

[12] M.A. Pund, S.R. Jedhao and P.D.Thakare,"A Role of Query Optimization in Relational Database," Int. journal of scientific engineering and research, vol. 2, June 2011.

[13] Chun–Nan Hsu and Craig A. Knoblock," Rule induction for Semantic Query Optimization,"Information science institute AAA1, Technical report on Knowledge discovery in Databases, 1994, pp. 311-317.

[14] Peter Paul Beran, Werner Mach and Ralph vigne, " a heuristic Query Optimization approach for Heterogeneous Environments," In proceedings of the 10th IEEE , ACM International Conference on cluster and cloud computing , 2010, pp. 542-546.

[15] Dunrenche and Karl Aberer," A Heuristic based approach to Query Optimization in structured document databases," In the proceedings of the IDEAS99 International Symposium, 1999.

[16] J.C. Freytag ," The Basic Principles of Query Optimization," European computer industry research centre, IFIP, 1989.

[17] M.L. Rupley, " Introduction to Query Processing and Optimization ,Indiana University,2008.

[18] Avi Silbershatz, Hank Korth and S. Sudarshan, "Database System Concepts,"4th edition ,Mc Graw hill , Technical Report,2002.

[19] M. Joseph, 'Optimization techniques for queries with expensive methods," ACM Transactions on Database systems, vol.23, June 1998, pp. 113-157.

[20] D.U. Weimin, "Query Optimization in Heterogeneous DBMS," In proceedings of the 18th VLDB conference vancover,1992.

[21] J.Grant, J. Gryz, J. Minker and L. Rashid,"Semantic Query Optimization for Object Databases," In ICDE, 1997, pp. 444-453.

[22] A. Swami and A. Gupta ," Optimization of Large Join Queries ," In proceedings of the ACM SIGMOD Conference, Chicago, June 1998, pp. 8-17.

[23] S. b. Yao, " Optimization of Query evaluation algorithms," ACM Trans. Database Systems, vol.4, June 1979, pp. 133-155.

[24] Sreekumar T. shenoy and Zehra Meral Ozsoyoglu," Design and Implementation of a Semantic Query Optimizer," IEEE Transactions on knowledge and Data engineering ,vol. 2, Sep. 1989.

[25] P.P.S. Chen and J. Akoka,"Optimal design of Distributed Information sysytems," IEEE Transactions Computing Database., vol. 12, 1980, pp. 1068-1080.

[26] A.K. Chandra and P.M. Merlin," Optimal implementation of conjunctive queries in relational databases," In proceedings of the 9th ACM Symposium on Theory of Computation, New York, 1977, pp. 77-90.

[27] B. Gavish and A. segev,"Query Optimization in distributed computer systems ," In management of distributed data processing, New York, 1982, pp. 233-252.

[28] W.W. Chu and P. Hurley, " Optimal Query processing for distributed database systems," IEEE Trans., Comput., 1982, pp. 835-850.

[29] J.J.King," A System for Semantic Query Optimization in Relatinal databases," In proceedings of 7th VLDB conference,pp. 510-517.

[30] C.Hsu and C.A. Knoblock ," Rule Induction for sematic Query Optimization ," Inproceedings of 11th International conference on machine learning,1994, pp. 112-120.