A review on Different Approaches of Query Optimization Technique

Ambarish R. Bhuyar [1]
ambarishbhuyar10@gmail.com
Apurva B. Parandekar [2]
apurvaparandekar07@gmail.com
Snehal R. Kamble [3]
srksipna@gmail.com
Department of Information Technology,
Sipna College of Engineering & Technology, Amravati, Maharashtra, India.

Abstract: Query optimization is the process of selecting the most efficient query evaluation plan from among the many strategies usually possible for processing a given query especially if the query is complex. It is a purpose of many relational database management systems. It is a key technology for every application from operational systems to data warehouse and from analytical systems to content management systems. The query optimizer experiments to dictate the most efficient way to implement a given query by examining the possible query plans. The performance of database systems is critically dependent upon the efficiency of Optimization techniques and Query Execution Operators. In this paper we present various Optimization techniques and Query execution Operators that helps to enhance the query performance.

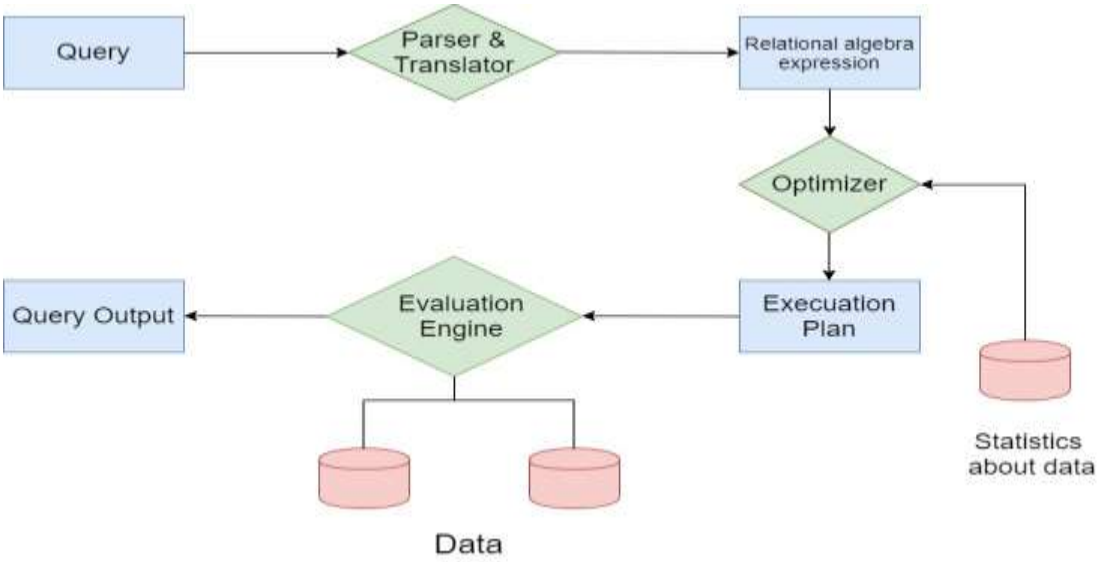
I. Introduction

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 [1]. Query Optimization [2] comes into play when we expect the system to construct a query evaluation plan. One aspect of optimization occurs at the relation algebra level, where the system attempts to find an expression that is equivalent to the given expression but more efficient to execute. Another aspect is selection a detailed strategy for processing the query, such as choosing the algorithm to use for executing an operation, choosing the specific indices to use and so on. The difference in cost (in terms of evaluation plan) between a good strategy and a bad strategy is often substantial, and may be several orders of magnitude. Hence it is worthwhile for the system to spend a substantial amount of time on the selection of a good strategy for processing a query, even if the query is executed only once. In general Query Optimization [3] 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. A desirable Optimizer [4] 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 [5] 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.

II. Query Flow

The first step in processing a query submitted to a DBMS is to convert the query into a form usable by the query processing engine. High- level query languages such as SQL represent a query as a string, or sequence, of characters. Certain sequences of characters represent different types of tokens such as keywords, operators, operands, literal strings. The primary job of the parser is to extract the tokens from the raw string of characters and translate them into the corresponding internal data elements for example relational algebra operations and operands and structures for example query tree, query graph. The last job of the parser is to check the validity and syntax of

the actual query string [1]. In second stage, the query processor applies instructions to the internal data structures of the query to convert these structures into complement, but more efficient representations. The rules can be based upon mathematical models of the relational algebra expression and heuristics, upon cost approximates of different algorithms or techniques applied to operations or upon the semantics within the query and the relations it necessitates. Selecting the absolute rules to apply, when to apply them and how they are applied is the function of the query optimization engine [6].



2.1 Steps in Query Processing

III. Query Optimization Principles

Query Optimization [7, 19] 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 [8] 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 [9] 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.

IV. Overview of Query Optimization Techniques

Query Optimization [13, 14, 21] 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 [15, 16] 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 [17, 18, 22] 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.

V. Approaches for Query Optimization

5.1 An Efficient Processing of Queries with Joins and Aggregate Functions - View Selection

In this technique, it proposes a new method for processing the queries including both joins and aggregate functions. The method performs grouping dimension tables with group-by conditions at first and then processes joins by using bitmap join indices [24]. It is very important to procedure efficiently expensive queries including joins and/or aggregate functions in data warehousing environment since there resides extensive volume of data and the procedure of these queries takes much time. This allows to process aggregate functions by accessing fact tables only, thus it can reduce the serious performance degradation of existing methods. In order to show the superiority of the method, they develop a cost model for both the proposed and the existing ones, and perform extensive simulations based on the TPC-H benchmark. They have proposed a technique that performs grouping of the dimension table in advance and removes the number of disk accesses in grouping the joined table. The dimension table involves the grouping attributes for aggregate functions and the fact table involves the aggregate attributes [24][25]. With the characteristics, grouping of the dimension table is performed by the proposed procedure which is quite smaller than the fact table. This approach is able to perform aggregate functions efficiently using the bitmap join indices developed for processing joins.

5.2 A Comprehensive Analysis of Materialized Views

Data in a warehouse can be perceived as a collection of materialized views that are generated as per the user requirements specified in the queries being generated against the information contained in the warehouse. User requirements and restraints frequently change over time, which may evolve data and view definitions stored in a data warehouse dynamically. The current requirements are changed and some novel and innovative requirements are added in order to deal with the latest business scenarios [25]. In fact, data conserved in a warehouse along with these materialized views must also be updated and maintained so that they can deal with the changes in data sources as well as the requirements stated by the users. Selection and maintenance of these views is one of the vital assignments in a data warehousing environment in order to provide optimal effectiveness by decreasing the query response time, query processing and maintenance costs as well.

This analysis has presented different approaches being proposed by different researchers to deal with the materialized views in data warehouse that is to say view adaptation & synchronization, view selection and view maintenance [26]. They have examined these techniques on various parameters and provided a comparative study in a tabular manner.

5.3 Algorithms for Materialized View Selection and Maintenance

The materialization of all views is practically impossible because of the materialized view storage space and maintenance cost constraint thus proper materialized views selection is one of the intelligent decisions in designing a data warehouse to get optimal efficiency. It presents a framework for selecting better materialized view so as to achieve the effective combination of good query response time, less query processing cost and low view maintenance cost in a specified storage space constraint [27]. Query frequency cost, query storage cost and query processing cost is included by the structure implementation parameters. The structure select the best cost efficient materialize views to optimize the query processing time thereby resulting efficient data warehousing system. This gives the suggestion regarding best view selection for materialization and the effective incremental batch method for materialized view maintenance.

The first approach is for materialized view selection, the essential constraints for materialize view selection are: query frequency, query processing time and storage space [28]. The presented recommended methodology determines which queries are more advantageous using combination of query frequency, processing and storage cost for the creation of materialized view so as to achieve the quick query processing time.

5.4 Materialized view Size Estimation using Sampling

Materializing these summary views can minimize query response time but as the number of views increases exponentially with the number of dimensions, all the views cannot be materialized. However under storage constraint, they must be able to select an optimal number of views to be materialized. To determine the views to be materialized, a Data warehouse administrator may not be able to afford the necessary time to govern the number of rows (size) in each view. Counting actual number of rows represent in each view takes considerable time. They explore the use of sampling to approximate the size of views [29]. It proposes a hybrid estimator that takes into account the degree of skew in the data and combines estimators to estimate the size of the view more accurately. The view size has been used by the proposed hybrid estimator in tpc-h benchmark and the results show better estimation results as compared to individual estimators.

In sequence to apply any problem solving approach, the first step is to know the parameters of the problem occurrence. To completely specify a materialized view selection problem (MVS), one must know the number of rows present in each view and access frequency of each view [29]. Determining access prevalence of each view is a managerial circumspection. The culmination of the contribution of this approach is to apply the concept of sampling based row estimation in a relational table to approximate the size of the views in a data cube taking into account the degree of data skew.

5.5 Comparison between techniques

Techniques	Description
View Selection	Materialization of all possible views is not recommended due to memory space and time constraints. It is reduce space and time.
View Adaption	View adaption problem one of the factors that contribute to the changes in a materialized view is rewriting of views that leads to changes in the original view definition itself.
View Synchronization	View synchronization technique changes the view definition when the structure of its base relations changes.
View Maintenance	It is incrementally updates a view by evaluating the changes to be incorporated in the view.

Table 5.1: Comparison between methods

VI. Conclusion

In this paper we have discussed about various Query Optimization techniques. 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 discussed an analysis of different technique being proposed by various researchers to deal with the materialized views in data warehouse namely- view adaptation, view synchronization, view selection and view maintenance. They are all schema object base views techniques. The view selection, view Adaption, view synchronization, view maintenance of to schema object base is one of the most important issues in designing a data warehouse. So as to achieve the best combination of good query response where query processing cost should be decrees in a given storage space constraints. The space constraint is the most important factor while selecting the views to be schema object base. We are taking different parameters for all techniques and analysis. By using these techniques reduce query execution cost, spaces reduce and it is also maintaining large data set.

References:

[1] Tejy Johnson and Dr.S.K.Srivatsa, "A Study on Optimization Techniques and Query Execution Operators That Enhances Query Performance ", in International Journal of Advanced Research in Computer Science, Volume 3, No. 3, May-June 2012.

[2] Abraham Silberschatz, Henry Korth, S. Sudarshan "Database System Concept". Query Optimization, pp 569.

[3] 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.

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

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

[6] Dhaval Patel, Pratik Patel, "A Review Paper on Different Approaches for Query Optimization using Schema Object base View", International Journal of Computer Applications (0975 – 8887), Volume 114 – No. 4, March 2015.

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

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

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

[15] 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.

[16] 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.

[17] 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.

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

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

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

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

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

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

[24] T.Nalini, Dr. A.Kumaravel, Dr.K.Rangarajan "A comparative study analysis of materialized view for selection cost" International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.1, February 2012.

[25] Garima Thakur, Anjana Gosain " A Comprehensive Analysis of Materialized Views in a Data Warehouse Environment" (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 5, 2011

[26] Deepika Kirti, Jaspreeti Singh "Assortment of Materialized View: A Comparative Survey in Data Warehouse Environment"International Journal of Computer Applications (0975 – 8887) Volume 96– No.7, June 2014.

[27] Ravindra N. Jogekar, Ashish Mohod, "Design and Implementation of Algorithms for Materialized View Selection and Maintenance in Data Warehousing Environment" International Journal of Emerging Technology and Advanced Engineering Volume 3, Issue 9, September 2013.

[28] S. Chen, X. Zhang, and E. Rundensteiner, "A compensation based approach for view maintenance in distributed environments", In IEEE transactions and data engineering, 18, 2006.

[29] Madhu Bhan, T.V.Suresh Kumar, K.Rajanikanth "Materialized view size estimation using sampling" IEEE International Conference on Computational Intelligence and Computing Research 2013 IEEE.