International Journal of Scientific Research in Computer Science and Engineering
Technical Paper Vol-2, Issue-1 E-ISSN: 2320-7639

# Improving Effectiveness of Query Optimizer

Jyoti Haweliya[1*], Ravinder Kaur Narang[2]

[1*]*Department of Computer Engineering, IET, DAVV, Indore, India, jyoti.samad@rediffmail.com*
[2] *Department of Information Technology, IET, DAVV, Indore, India, ravinder11narang@yahoo.com*

**Available online at www.isroset.org**

Received: 24 Dec 2013 Revised: 08 Jan 2014 Accepted: 20 Jan 2014 Published: 28 Feb 2014

**Abstract—** Query Optimization is important tasks in Relational DBMS. Given a query, there are many plans that a database management system (DBMS) can follow to process it and produce its answer. All plans are equivalent in terms of their final output but vary in their cost, i.e., the amount of time that they need to run. Process of finding good evaluation plan is called Query Optimization .The plan is formed from different combination of operators and the way these operators are implemented in database affects the cost of query. However, the literature do not attempt to give us comparison of different implementation algorithms for operators and to what particular type of plan a Query Optimizer should select for a particular situation. Therefore it is the purpose of this paper to understand and develop a comparison among them by taking example of a simple query. Further on basis of our analysis, we found some proposals that will help Query Optimizer to rule out good plans from bad plans in effect improving effectiveness of Query Optimizer.

**Keywords-** Query Optimizer, Prefer Pipelining, Sequential Access

## I. INTRODUCTION

Johann Christopher [1] describes the path that a query traverses through a DBMS until its answer is generated. The query is first parsed and its validity is checked .Then query is passed to query optimizer .The query optimizer generates all alternative plans and plans are then evaluated to find the cost for each plan and finally the best plan is selected Then the best plan passed to Query Processor for executing against database. This paper focus on the particular query optimization process. Ramkrishnan and Gehrek [9] described that the process of query optimization involves two basic steps as first enumerating all alternative plans and the second is to evaluate costs to select the best plan. The different plans are possible for executing query .Each plan is formed by some rearrangement of operators. As for example Consider a query having select and join both type of operations to be performed .So, First plan can be as selection operation performed first and then join performed on the results, Second plan can be as joined performed first on the join attribute and then selection done on the result, Another plan can be the way selection operator implemented as selection may be implemented as sequential or using btree or hashing etc.Then after evaluating all different possible plans ,the query optimizer estimate the cost of each enumerated plan and choosing the plan with lowest cost .For costs evaluation there are various steps performed by an optimizer as reading input tables which can be done using sequential,iteration,indexing etc.Then optimizer writes the results of search or read. Further if results are required in sorted form so, sorting step also adds to the costs. Each of above step incurs a cost, reading tables, writing temporary results and then if sorting is required, then results are sorted.

*Corresponding Author: Jyoti Haweliya*

Yannis E. Ioannidis [3] described that the costs for each above can be calculated using standard cost functions. So, the way an operator is implemented affects the costs.

Costs can be calculated by various costs evaluation functions. The way these different implementation algorithms are presented makes it difficult to compare and analyze them. So; the goal of this paper is to provide the comparison of some algorithms by taking an example of a simple query and to analyze all different available options on it. Further we found some proposals will help query optimizer to take decisions as to which algorithm will be best in what situation.

## II. LITERATURE REVIEW

Johann Christophe[1] describes wide variety of Query Optimization techniques .The way in which these techniques are presented gives their output same but differs in the way they optimize the query .Different techniques address different order in which to execute the operations. For each Technique an evaluation plan is constructed and it is the Query optimizer's responsibility to create evaluation plan Ramkrishnan and Gehrek [9]described that evaluation plans are formed from different operators and the operators can be implemented using different algorithms .So the implementation algorithms affects the cost of complete evaluation plan. After creating an evaluation plan, Query Optimizer makes use of System catalog and implementation information to evaluate costs [3].System catalog in database contains the complete data and metadata for the table .So, Query Optimizer makes use of data as how many pages, rows the table consists of. What type of access structure is there on the Index, no of attributes, logical and physical structure etc

 

Literature also gives us various cost evaluation functions to evaluate costs of different evaluation plans. [4]. There are various advanced ways of Query Optimization [5] that researchers have proposed over the past years as Semantic Query,Nested Query,Mutiple Query, Dynamic ,parametric etc. Semantic Query Optimization [6] is a form which relies on rewriting a given Query semantically equivalent Ones. This makes use of heuristics and rule based approach for rewritings. Further Optimization is on Global Query Optimization [7]. This presents work on queries that become available for optimization at the same time, from multiple concurrent users, or embedded in a single program.So, here instead of optimizing a single Query, one may be able to find a good plan for executing group of queries. Further work on Parametric and Dynamic Query Optimization [8], propose using the actual parameter values at run time and simply pick the plan that is optimal with little or no overhead.Futhermore, there are much interesting work done in the areas like rule based optimization, nested queries, aggregate query optimization etc their further details can be provided in the references provided. But goal of this paper is to present issues related to simple individual queries and how query optimizer works for simple queries

## III. METHODOLOGY

As described by Ramkrishnan and Gehrek query optimizer while evaluating the query finds out all alterative options to process the query. For a query there can be enormous number of plans that can be possible to process it, but for calculating costs Optimizer does not evaluate all generated plans but some subset of plans. Based on rules as defined by query optimization algorithm optimizer is able to eliminate bad plans. Costs Evaluation Functions given by Korth [10] are as follows

NPages (A) =No of Pages of Relation
a Sequential Scan: NPages
Hash Index(r)/b (I)
Btree: d (I) +p(r)*s/O (log (n))
Tuple: Npages1+ (tupleSize*Npages2)
Page: Npages1+ (Npages1*Npages2)
Nested: M+N. [M/B]
Merge: sort(r) +sort(s) +p(r) +p(s)

Here we present a methodology for Query Optimization. The steps are approximation of what actually system does to evaluate Query, find costs and select best plan. By taking example of Single Relation Query, we aim at describing the complete optimization procedure an optimizer would follow to execute it. Example Consider following query:

Select first.IntId, second. Status, second.refId from User first, User second where first.IntId=second.refId.The evaluation plan can be constructed as:

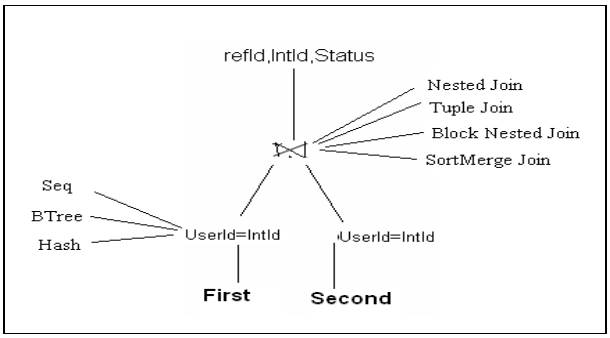


Figure 1:Evaluation plan for Query

But there are several other plans are possible like performing self join or taking cross product and then performing selection operation or first selection and then the join. Here we mention a plan which shows that any type of search is possible on selection operation and any type of join is possible. But which combination is best can be found by utilizing costs and system catalog information. As for example for this particular plan we consider no of rows for User table as 1000 and by utilizing following Literature Key points we can find the best combination of join and search to evaluate cost. On basis of key points and cost calculation methods made available to us by various earlier Literature and papers, we are able to prepare following comparison chart for various combinations of operators. By looking at this chart, we conclude that for a query which contains equality as a conjunct, Hash Index (for search) in combination with Sort-Merge join is the best Evaluation plan.

TABLE I: COMPARISON TABLE

| Plan | Page | Tuple | Nested | Merge |
|---|---|---|---|---|
| Sequential | 61 | 120 | 29 | 21 |
| BTree | 43 | 102 | 11.15 | 3.5 |
| Hash Index | 41 | 121 | 9 | 1.5 |

Fig1.2 Comparison Chart

*A Proposal : Cross product should never be formed*
Results of a cross-product is typically much larger than result of a join, Joins have therefore received a lot of attention. Example cost of result can be approximated by the fraction of tuples that will appear in the result and fraction of tuples depends on the reduction factor as

$$\frac{\text{NPages}*(\text{size of attribute})*\text{RF}}{(\text{size of tuple})\quad(\text{NK})}$$

RF:-Fraction of tuples that satisfies a given conjunct Cross Product consists of almost entire table's rows as compared to joins which retrieve only the matching rows in the result.

 

*B Proposal : Give preference to Sequential access over Unclustered Btree*

Sequential access scans entire file .Example Table Employee contains 1000 pages and selection is of the form name<'C%'. We estimate that roughly 10% of employee tuples are in the result .This is total of 100 pages. or 10,000 tuples so, cost of sequential scanning Employee would be 1000I/Os depends on the no. of pages. Whereas consider the case of unclustered Btree Index .here each tuple could cause us to read a page .so in the worst case we could have up to 10,000I/Os.

*C Proposal : Prefer Pipelining over Materialization*

When Query composed of multiple operators, result of an operator is to be passed to other Operator up in the hierarchy. As above mentioned when costs of result is calculated, step to write intermediate results also affects the costs.So,if result of an operator is pipelined rather than materializing a huge costs can be saved. Example costs of writing intermediate results is given by following formula

$$\frac{\text{NPages}*(\text{size of attribute})*\text{RF}}{(\text{size of tuple})\ \ (\text{NK})}$$

This cost adds in the final result. In pipelining intermediate results are not written to temporary tables

*D Proposal : Make Use of Partitioning in joins (if possible).*

Implementation technique like Simple Nested loop for joins enumerate all tuples of tables to be joined and discard the tuples that do not meet the join condition. Whereas the algorithms like Index Nested Loop Join makes use of partitioning, tuples in the two relations can be thought of as belonging to partitions, such that tuples in the same partitions can join with each other and for each tuple in one relation, it uses an index on second relation to locate tuples in same partition. Thus only a subset is compared entire relation is never compared. Example Simple Nested Loop is a tuple-at-a-time evaluation. So, if Employee table consists of 1000 pages and each page is of 100 rows which is to be joined with 500 pages of Manager Table .Then cost of Simple Nested Loop will be 1000+(1000*100*500) I/Os which gives costs as 1000+(5*10e7)I/Os. As compared to Index Nested Loop which utilizes an index on second Relation, consider example of hash index, so costs goes down to 100*1000*(1+1.2)which is equal to 221,000I/Os. So, even if go for any type of indexing Index Nested Loop performs much better than Simple Nested Loop.

*E Proposal : Sort-Merge Join Vs Block Nested Loop Join*

Sort-Merge Join as compared to Block Nested Loop makes use of Partitioning and but do not rely on pre-existing Index. A Sort-Merge Join sort the two relations on join attribute and merge phase begins with scanning each tuple of each relation. Cost of sorting is O (MlogM) and cost of merging is M+N. so the total costs becomes:

$$O(M\log M)+O(N\log N)+M+N.$$

Example consider 1000 and 500 pages of Employee and Manager respectively. cost of sort-Merge Join becomes 4000+2000+1000+500=7500I/Os Consider block Nested Loop , which relies on utilizing available number of buffer pages ,If buffer pages are less to hold entire relation ,so break relation into blocks that can fit into available blocks. costs is given as

$$M+(N*[M/B])$$

where M,N are no of I/Os to scan the two relations and B is available number of buffer pages. So, if we consider no of buffer pages available as 35 so the cost for above Employee and Manager relation becomes becomes 15000I/Os , which is not better than Sort-Merge Join but if we increase buffer pages to 300 so the costs of block nested loop drops to 2500I/Os which is better than sort-Merge Join,Finally available buffer pages affects choice of algorithm.

## IV. CONCLUSION

We analyzed that query consists of many operations and each operation can have many execution options, and each execution options can be evaluated by a cost fuctions, by which we can find some best evaluation strategy. We derived comparison chart and Best practices which will help query optimizer to make decisions to select efficient plan and eliminate several bad plans that are generated. Further in some cases chosen optimization plan may not be the optimal (best) strategy – it is just a reasonably efficient strategy for executing query.

## REFERENCES

[1]. Johann Christopher Freytag, "Basic Principles of Query Optimization", March **1989/09**, CA, pp **801-807.**
[2]. Yannis E. Ioannidis, University of Wisconsin, "Query Optimization", June **1982**.
[3]. Visshy Posala, Bell Labs, Query Optimization, April **1992**
[4]. S. Mein, "Principles of Query Optimization", **1990**.
[5]. L. Vieille , "Advanced Query Optimization", March **1989**
[6]. C. White., Principles of Semantic Query , **1986**
[7]. G.von Bultz, Translating and Optimizing Global Queries, **1987**
[8]. W. Wang , Optimizing Dynamic AND Parametric Queries, **1985**
[9]. Ram Krishnan and Gehrek, Introduction to Relational Database
[10]. Silberschatz, Korth, Sudarshan, Introduction to Database concepts

 