International Journal of Advanced Computer Research
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
Volume-5 Issue-19 June-2015

A Simplistic Mechanism for Query Cost Optimization

Debopam De*, Deblina Banerjee, Sneha Mukherjee and Jayati Ghosh Dastidar
St. Xavier's College, Kolkata

Received: 06-May-2015; Revised: 03-June-2015; Accepted: 7-June-2015

Abstract

The cost of a database query can be optimized so that a more efficient query can be generated. However not many tools are available which work independently to optimize the cost of a query. This article is based on a tool that we developed to serve the purpose. It uses an Oracle Database and Linear Programming Problem concepts to evaluate the optimum cost of the query and compares it with the original cost of the query that the user gives. The optimum cost can also be used to devise a new query which serves the same purpose as the original one and with reduced cost.

Keywords

Database statistics, plan table records, cost retrieval, cardinality retrieval, bytes retrieval, optimization of cost, cost comparison, simplex.

1. Introduction

Optimization is the act of obtaining the best result under given circumstances. In design, construction and maintenance of any real-life system, we have to take many technological and managerial decisions at several stages. The ultimate goal of all such decisions is either to minimize the effort required or to maximize the desired benefit. The effort required or the benefit desired in any practical situation can be expressed as a function of the conditions that give maximum or minimum value. As a reference we can see the graph in Figure 1. Here the point x* corresponds to the minimum value of the function f(x), the same point also corresponds to the maximum value of the negative of the function, -f(x). Thus without loss of generality, optimization can be taken to mean minimization since the maximum of a function can be found out by seeking the minimum of the negative function. The project cost evaluation and optimization tool uses the cut edge technology of Cost-Based Optimization available in versions of Oracle (Oracle 7.0 onwards). The main theme of our paper is to develop software that takes as input a query and analyses its execution plan. After the analysis is done on the query, the total cost, bytes consumed when the query is being executed, the I/O cost and the cardinality of each and every statement of the query is extracted. These factors are the pillars on which cost depends. We will attempt to optimize the cost of a query execution and our algorithm finds out an optimized cost wherever is possible (there are queries for which no further optimization is required). As the cost and other factors are retrieved, these factors are used to form equations and in-equations of the Simplex Method [9, 10]. The ultimate value of the objective function (formed by the cost only) tells us whether any further optimization is required or not and if so, then what can be the optimized value of the Objective Cost Function.

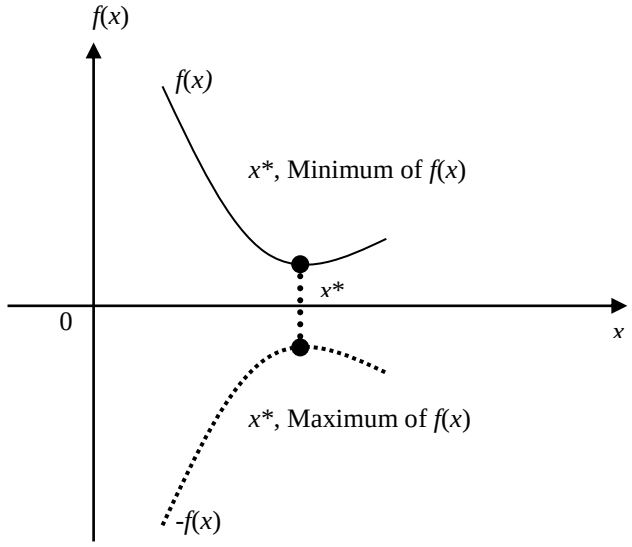


Figure 1: Minimum of *f*(*x*) is same as maximum of –*f*(*x*)

*Author for correspondence

There are two types of optimization models, viz. Rule Based Optimization (RBO) and Cost Based Optimization (CBO). The RBO uses a set of rules to determine how to execute a query. If an index was available on a table, the RBO always uses the index. There are some cases where the use of an index slows down a query. RBO, armed with its set of discrete rules, does not always make great decisions. The biggest problem with the RBO is that it does not take the data distribution into account. The RBO always uses an index if present, because it is so designed. In contrast, the CBO uses statistics about the table, its indexes and the data distribution to make better informed decisions. The CBO uses database statistics to generate several execution plans, picking the one with the lowest cost, where cost is dependent on system resources required to complete the operation. Basically, the RBO used a set of rules to determine how to execute a query. As it turns out that this is simpler to implement but not the best strategy. RBO was supported in earlier versions of Oracle. (SQL Server supports table hints which in a way can be compared to RBO, as they force the optimizer to follow a certain path). Motivation behind CBO is to come up with the cheapest execution plan available for each SQL statement. The cheapest plan is the one that will use the least amount of resources (CPU, Memory, I/O, etc.) to get the desired output. This can be a daunting task for DB engine as complex queries can generate thousands of possible execution paths, and selecting the best one can be quite expensive. CBO is supported by most databases including Oracle, SQL Server, etc.

2. Literature Review

A considerable amount of research has been done on query optimization. The author in [1] has presented a generalised overview of query optimization in relational database systems. The paper discusses the need and modalities of query optimization in relational database models. The ways and means of query optimization for a database containing multimedia repositories has been discussed by the authors in [2]. The authors in [3. 4, 5, 6] have discussed the theoretical aspects of query optimization in their respective text books. The query optimization principles as implemented in the relational database Oracle have been discussed in [7] and [8]. The authors of the paper [11] present an approach using object-oriented databases under inheritance that permits to enrich technique of query optimization existing in the object-oriented databases. Their experimental study shows performance of query after implementation of inheritance method using relational as well as object oriented database. The Oracle White Paper [12] has discussed the mechanics of the explain Plan Table. The purpose of the Oracle Optimizer is to determine the most efficient execution plan for our queries. It makes these decisions based on the statistical information it has about the data and by leveraging Oracle database features such as hash joins, parallel query, and partitioning. The explain plan is by far the most useful tool at our disposal when it comes to investigating why the Optimizer makes the decisions it makes. By breaking down the explain plan and reviewing the four key elements of: cardinality estimations, access methods, join methods, and join orders; we can determine if the execution plan is the best available plan. The authors in [13] discuss query optimization for queries involving large join operations. It combines heuristics and combinatorial techniques for such queries. The authors in [14] have presented a novel technique for query optimization using the Genetic Algorithm. Their paper proposes a set of mutation strategies that are conscious of the nature of execution plans, and show that their combination, or their use with crossover operations, yields faster convergence and better plans than other more trivial mutation strategies. In addition, their mutations do not require repair operations after applying them.

The use of simplex method [9, 10] as a possible cost optimization technique for queries is yet to be fully explored. Thus we tried to apply this technique in trying to optimize the cost and met with reasonable success.

3. Methodology

Our attempt at cost optimization begins by using the Explain Plan feature of Oracle. The Explain Plan statement displays execution plans chosen by the Oracle optimizer for Select, Update, Insert and Delete statements. A statement's execution plan is the sequence of operations Oracle performs to run the statement. The Explain Plan results let us determine whether the optimizer selects a particular execution plan, such as, nested loops join. It also helps us to understand the optimizer decisions, such as why the optimizer chose a nested loop join instead of a hash join, and lets us understand the performance of a

query. The Explain Plan method does not require the query to be run. Thus saving a lot of time and resource compared to Autotrace. We next retrieve the cost constraints such as Bytes Constraint, I/O Cost Constraint and Cardinality Constraint from the output of the Explain Plan. These constraints are then formulated in the form of linear in-equalities. Finally the objective cost function is generated which represents the total cost. The final step involves finding an optimum solution to the linear programming problem that has been generated through the plan. For this we use the Simplex Method for Linear Programming Problem. The output gives us the optimized cost. The Figure 2 below depicts the flow of our proposed method:

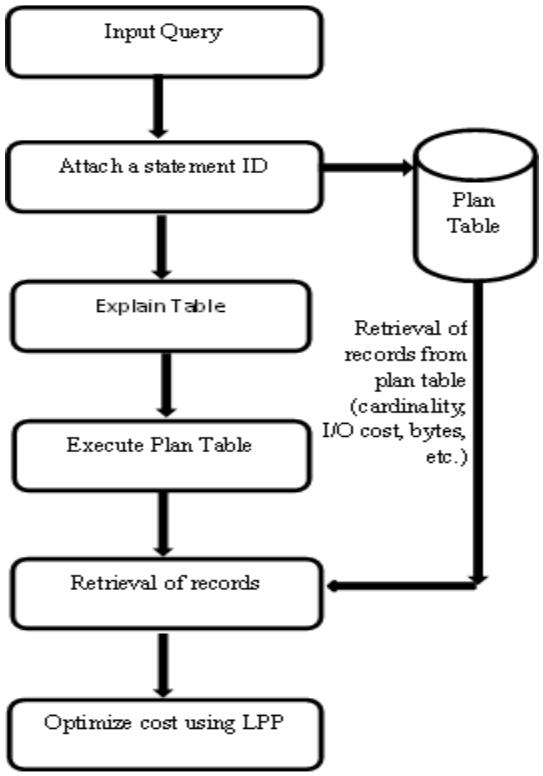


Figure 2: Optimization Process

The following algorithm summarises our approach to cost optimization:

Step 1: Begin
Step 2: Set a unique statement_id for a particular SQL statement and explain it through Oracle (Oracle 7.0 onwards).

Step 3: Any SQL statement, be it a DDL, DML, DQL or DSL has some cost cardinality and some byte consumption. Retrieve the cost, cardinality and the bytes used to execute the SQL query.
Step 4: Once the cost and the bytes are determined the inequations are formed and the objective function based on cost is retrieved as follows:
Minimize $Z = C_1x_1 + C_2x_2 + \ldots\ldots\ldots + C_nx_n$
Subject to the constraints:
Bytes Constraint:
$B_1x_1+B_2x_2 + \ldots + B_nx_n \geq (B_1+ B_2 + \ldots + B_n)$
I/O Cost Constraint:
$I/O_1x_1+I/O_2x_2+\ldots+I/O_nx_n \geq (I/O_1+I/O_2+ \ldots+ I/O_n)$
Cardinality Constraint:
$Cr_1x_1+Cr_2x_2+\ldots+Cr_nx_n \geq (Cr_1+Cr_2+\ldots+Cr_n)$
where, C_j, B_j, I/O_j and Cr_j are Cost, Bytes, I/O Cost and Cardinality constants retrieved, and are about the query for which cost estimation and optimization is done.

The program used for simplex algorithm, takes as input the objective functions and the subjective functions and evaluates the cost in every step and displays it. Thus, we can see whether the costs given in the format of equations are optimized or not.
We have used the Simplex Algorithm to optimize the cost. Following is the Simplex algorithm used to solve the Linear Programming Problem formulated above:

Simplex Algorithm:
Step 1: Modify the constraints so that the RHS of each constraint is nonnegative (This requires that each constraint with a negative RHS be multiplied by -1. Remember that if we multiply an inequality by any negative number, the direction of the inequality is reversed!). After modification, identify each constraint as a $\leq$, $\geq$, or = constraint.
Step 2: Convert each inequality constraint to standard form (If constraint is a $\leq$ constraint, we add a surplus variable s_i; and if constraint is a $\geq$ constraint, we subtract an excess variable e_i).
Step 3: Add an artificial variable a_i to the constraints identified as $\geq$ or = constraints at the end of Step 1. Also add the sign restriction $a_i \geq 0$.
Step 4: If the Linear Programming Problem is a maximisation problem, add (for each artificial variable) $-Ma_i$ to the objective function where M denote a very large positive number.

Step 5: If the LP is a minimisation problem, add (for each artificial variable) Ma_i to the objective function.

Step 6: Solve the transformed problem by the simplex method. Since each artificial variable will be in the starting basis, all artificial variables must be eliminated from the rows before beginning the simplex. Now in choosing the entering variable, we have to remember that M is a very large positive number.

If all artificial variables are equal to zero in the optimal solution, we have found the optimal solution to the original problem. If any artificial variables are positive in the optimal solution, the original problem is infeasible!

4. Results and Discussion

We have developed a Graphic User Interface using Visual basic which will enable one to input the query, analyse it, determine the cost and then optimize it. The Optimization module has been developed using the C programming language. Figure 3 shows the user interface that has been designed by us through which the query will be entered.

Case Study: We take up the following query and analyse it through the tool developed by us:
SELECT *ENAME, EMPNO, DNAME, LOC*
FROM *EMP, DEPT*
WHERE *EMP.DEPTNO = DEPT.DEPTNO*;

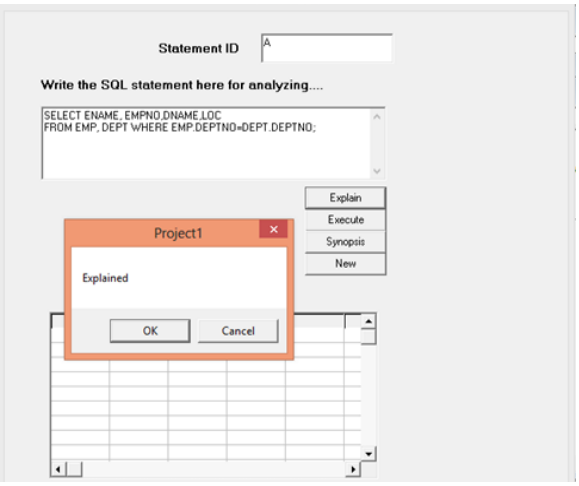


Figure 3: Graphic User Interface

By clicking on the Explain button, the supplied query will be "EXPLAINED" in Oracle and a message will appear in the model as "Explained". Oracle maintains a PLAN_TABLE and stores the related records, mentioned above in statement no 1, in that PLAN_TABLE against the supplied STATEMENT_ID. Subsequently, if we "Execute" the same query Oracle will supply the relevant records from the PLAN_TABLE and that will be displayed through the MSFlexGridControl provided in the model.

ID	PARENT_ID	OPERATION	OBJECT_NAME
0	0	SELECT STATEMENT	0
1	0	HASH JOIN	0
2	1	TABLE ACCESS	DEPT
3	1	TABLE ACCESS	EMP

POSITION	BYTES	COST	I/O_COST	CARDINALITY
3	392	3	3	14
1	392	3	3	14
1	72	1	1	4
2	140	1	1	14

Figure 4: Outcome of Explanation of query

Figure 5 shows the estimated cost taking into account all the factors described before. The coefficients for the different variables are obtained from the result of the query explanation (Figure 4 and Figure 5).

Figure 4 shows that there are four processes related with the above SQL statement. The Processes are identified by the Process_IDs as 0,1,2,3. The corresponding Parent_IDs are NULL, 0, 1, 1. For avoiding the syntactical malfunctioning the NULL values have been converted to 0. If the above SQL query is analysed, it will be observed that the statement consists of SELECT statement (first operation of the program). The records are to be retrieved from two different tables. But the two tables have been joined through an INNER JOIN. Joins are stored as the HASH JOIN through Oracle. For this reason, the second operation that is displayed is "Hash Join". The next two operations are "TABLE ACCESS" related to two tables "EMP" and "DEPT" which are displayed in the OBJECT_NAME attribute. Thus there are four operations in total that would have to be executed by the underlying database application (Oracle in this case).

At the time of execution of the query the EMP table contained 14 tuples (Cardinality 14) and the DEPT table contained 4 tuples (Cardinality 4). For execution of each process of the above SQL statement, the corresponding Cardinality that is to be traversed is displayed under the cardinality attribute.

They are 14, 14, 4 and 14 respectively. The last two tuples show the exact cardinality of the two relations (tables). Other attributes like Bytes, Cost and IO Cost determine the respective resource consumption of the given SQL statement.

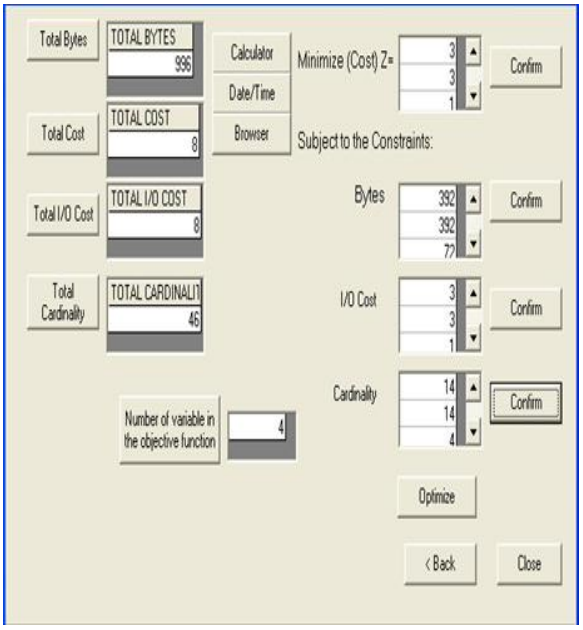


Figure 5: Cost estimation of query

Here the total of all the constraints including the Cost which is to be minimized is obtained and the linear programming problem is formulated. Three constraints viz. Byte Constraint (no. of bytes retrieved), I/O Constraint (no. of I/O operations – disk I/O) and Cardinality Constraint (no. of tuples retrieved) have been considered. The formulated Linear Programming Problem is as follows:

Minimize (Cost) $Z = 3x_1 + 3x_2 + 1x_3 + 1x_4$
Subject to the constraints:
Byte Constraint:
$392x_1 + 392x_2 + 72x_3 + 140x_4 \geq 996$
I/O Constraint:
$3x_1 + 3x_2 + 1x_3 + 1x_4 \geq 8$
Cardinality Constraint:
$14x_1 + 14x_2 + 4x_3 + 14x_4 \geq 46$
$x_1, x_2, x_3, x_4 \geq 0.$

Figure 6 represents the module of simplex where the actual optimization is done involving the above mentioned equations.

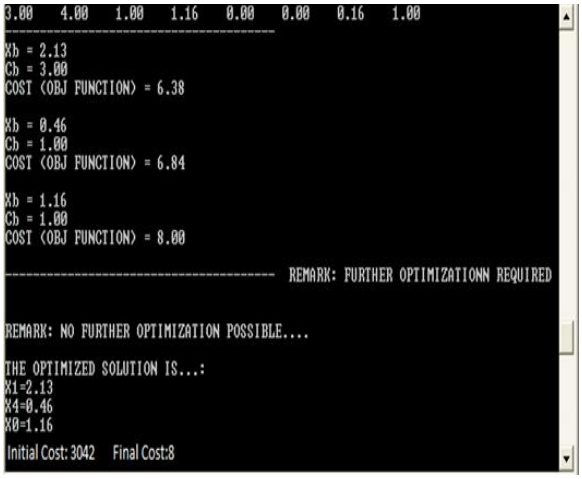


Figure 6: Screenshot of LPP solution

The initial cost function has a value 3042.00, whereas post optimization the computed cost is 8.00. This shows that there is further scope of optimising the query.

Figure 7 shows the graphical representation of the cost optimization process. It can be seen that the cost has become optimized when it acquires a value 8.00. The implication of this result is that the given query has further scope of optimization.

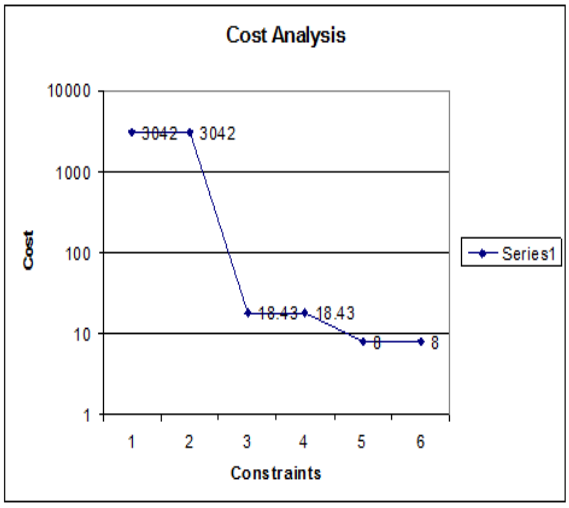


Figure 7: Graph to represent cost optimisation

Table 1 below shows the behaviour of 4 different queries. The total cost pre and post optimization has been shown.

Table 1: Comparative results of different queries

Query no.	Query	Pre-optimized Cost	Post-optimized Cost
1	select price from book,publish where book.id = publish.book_id;	558	48
2	select price from book Minus select b1.price from book b1,book b2 where b1.price < b2.price;	206	14
3	select no_of_publicaions from author a,book b,publish p where a.id = p.auth_id and b.id = p.book_id and price > 200**;**	336	28
4	select price from book b,publish p where b.id = p.book_id and auth_id = 'A001';	210	19

5. Conclusion and Future Work

This tool is user friendly software which is specially designed for providing an aid to the DBMS programmer to ascertain the COST of an SQL query. It will be a helpful tool to find out the Total Cost which will be incurred at run time because of utilization of System Resources. It is known that the cost and profit are dual of each other. So, if cost can be minimized (Optimized) then obviously, Profit can also be maximized (Optimized). The tool can also be used to find out whether a query is optimal or not.

Attempts can be further made to develop a design pattern of those un-optimized SQL queries so that they may attain the total cost that will be generated through this model.

References

[1] Victor Muntes Mulero, Josep Aguilar Saborit, Josep Lluis Larriba Pey y Calisto Zuzarte, "A Study of Execution Plan Aware Mutations for Genetic Cyclic Query Optimization", XV Jornadas De Paralelismo Almeria, September 2004.

[2] An Oracle white Paper, "The Oracle Optimizer, Explain the Explain plan", May 2011.

[3] Arun N. Swami, "Optimization of large join queries: Combining heuristic and combinatorial techniques", In SIGMOD Conference, ACM Press, pages 367-376, 1989.

[4] Abhijit Banubakode, G. S. Mate, "Query Optimization and Execution Plan Generation in the ObjectOriented Databases under Inheritance", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 12, December 2014, pages 963-969.

[5] Surajit Chaudhuri, "An overview of query optimization in relational systems", PODS '98 Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of database systems, pages 34-43.

[6] Surajit Chaudhuri, Luis Gravano, Method for cost-based optimization over multimedia repositories, Patent no. US 5806061 A.

[7] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Database System Concepts", McGraw-Hill International Edition, 5th edition.

[8] Thomas Connolly and Carolyn Begg, "Database Systems – A practical approach to Design,Implementation and Management", McGraw-Hill International Edition, 5th edition.

[9] Arun K. Majumdar and Pritimoy Bhattacharyya, "Database Management Systems", Tata McGraw-Hill Education Private Limited, New Delhi, edition: 29th reprint, 2011.

[10] Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, "Database Systems: The Complete Book", Prentice Hall Upper Saddle River, New Jersey 07458.

[11] "Oracle9i, Database Performance Tuning Guide and Reference", Release 2 (9.2), October 2002, Part No. A96533-02.

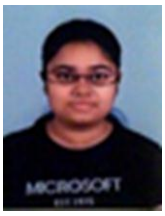
[12] Jonathan Lewis, "The Expert's Voice in Oracle – Cost-Based Oracle Fundamentals", Apress, 2006.

[13] J. A. Nelder, R. Mead, "A Simplex Method for Function Minimization", Oxford Journal – Science & Mathematics, Computer Journal, Volume 7, Issue 4, pages 308-313.

[14] George B. Dantzig, Alex Orden, Philip Wolfe, "The Generalized Simplex Method for minimizing a linear form under linear inequality restraints", Pacific Journal of Mathematics, Vol. 5, No. 2 October, 1955, pages 183-195.

Debopam De is currently an under graduate student of St. Xavier's College, Kolkata under the University of Calcutta in the department of Computer Science. His area of interest is Database Management System.

Email: debopamde94@gmail.com



Deblina Banerjee is currently an under graduate student of St. Xavier's College, Kolkata under the University of Calcutta in the department of Computer Science. Her area of interest is Computer Programming.

Sneha Mukherjee is currently an under graduate student of St. Xavier's College, Kolkata under the University of Calcutta in the department of Computer Science. Her area of interest is Web Page Designing.

Jayati Ghosh Dastidar completed her Master in Engineering (IT) from the WBUT, Kolkata. She is currently an Assistant Professor in the Department of Computer Science, St. Xavier's College (Autonomous) under the University of Calcutta, Kolkata, India.