Analysis of Query Optimization Techniques in Databases

Jyoti Mor
M. Tech Student, CSE Dept.
MRIU, Faridabad

Indu Kashyap
Assistant Professor, CSE Dept.
MRIU, Faridabad

R. K. Rathy, PhD.
Professor, CSE Department
MRIU, Faridabad

ABSTRACT

Query optimization in databases has gain a lot of importance in recent years. In this paper, we have analyzed different techniques of query optimization in relational databases and compared their performance. We have covered the techniques which use different methods for query representation.

Keywords
query; optimization; graph; tableaus; aggregate

1. INTRODUCTION

The databases are the best way of storing and viewing the data in form of tables. Databases are most useful in representing data in an organized manner. The query optimization in databases has gained equal importance as it is very important to reduce the size, memory usage and time required for any query to be processed. This improves significantly the performance of the database. Databases can be classified according to their organizational approach. The most popular approach is relational database, a tabular database in which data is defined so that it can be reorganized and accessed in a number of different ways. Another approach that has become popular now-a-days is object database. An object database applies the concept of object oriented programming. Object database corresponds with the data defined in object classes and subclasses. A database management system (DBMS) is software designed to assist in maintaining and utilizing large collections of data. The need for such systems, as well as their use, is growing rapidly. By storing data in a DBMS rather than as a collection of operating system files, we can use the DBMS's features to manage the data in a robust and efficient manner. This paper deals with relational databases.

2. QUERY OPTIMIZATION

Query optimization refers to the process of executing a query efficiently. This requires how to fire a given query such that it takes minimum number of operations and the memory space. It is the most important part of the query evaluation process. Query optimization is a function in which multiple query plans for satisfying a query are examined and a good query plan is identified. There is a trade-off between the amount of time spent figuring out the best plan and the amount running the plan. The resources which are considered for costing are CPU path length, amount of disk buffer space, disk storage service time, and interconnect usage between units of parallelism. The set of query plans examined is formed by examining possible access paths and various relational table join techniques. These plans are generated by the parser while parsing the query. The search space can become quite large depending on the complexity of the SQL query [1, 10].

2.1. Queries

A query is a language expression that describes data to be retrieved from a database. In the context of query optimization, it is often assumed that queries are expressed in a content-based manner, giving the optimizer sufficient choices among alternative evaluation procedures. Queries are expressed in several forms or settings. The most important application is that of direct requests by end users who need information about the structure or content of the database. Another application of queries occurs in transactions that change the stored data based on their current value [9]. Query like expressions can be used internally in a DBMS, to check access rights, maintain integrity constraints, and synchronize concurrent accesses correctly. For Example, SQL query to view students roll no, students name and total marks of those students who have scored more than 300 marks from the Student database would be as –

```
SELECT Rollno, Name, TotalMarks
FROM Student
WHERE TotalMarks>300
```

2.2. Optimization Objectives

Objective of optimization process should be either to maximize the output for a given number of resources or to minimize the resource usage for a given output. Query optimization tries to minimize the response time for a given query language and mix of query types in a given system environment [8]. The total cost to be minimized is sum of following costs:
Communication Cost: Cost of transmitting data from the site where they are stored to the sites where computations are performed and results are presented.
Secondary Storage Access Cost: The cost of loading data pages from secondary storage into main memory.
Storage Cost: The cost of occupying secondary storage and memory buffers.
Computation Cost: The cost for using the central processing unit (CPU).

3. OPTIMIZATION TECHNIQUES

Different query optimization techniques are defined by different people. These techniques are unique in their own representation and method.

3.1 Optimization using Query Graph

Query Graphs are used in query optimization for the representation of queries or query evaluation strategies. Two classes of graphs can be distinguished: object graphs and operator graphs [7].
Nodes in object graphs represent objects such as variables and constants. Edges describe predicates that these objects are to fulfill [12]. Operator graphs describe an operator-controlled data flow by representing operators as nodes that are connected by

edges indicating the direction of data movement [11]. Query graphs have many attractive properties. The visual presentation of a query contributes to an easier understanding of its structural characteristics.

Example 1: Consider two relational schemas.

Students (Rollno, Name, Address) and
Student_Marks (Rollno, English, Math, Science, TotalMarks, MarksPercentage)

Assume that the user wishes to view "Name, TotalMarks and MarksPercentage" of the students who have got TotalMarks greater than 235."

The query for this would be:

SELECT Name, TotalMarks, MarksPercentage
FROM Students, Student_Marks
WHERE Students.Rollno=Student_Marks.Rollno and (TotalMarks>235)

The query graph for this query is shown in fig 1.

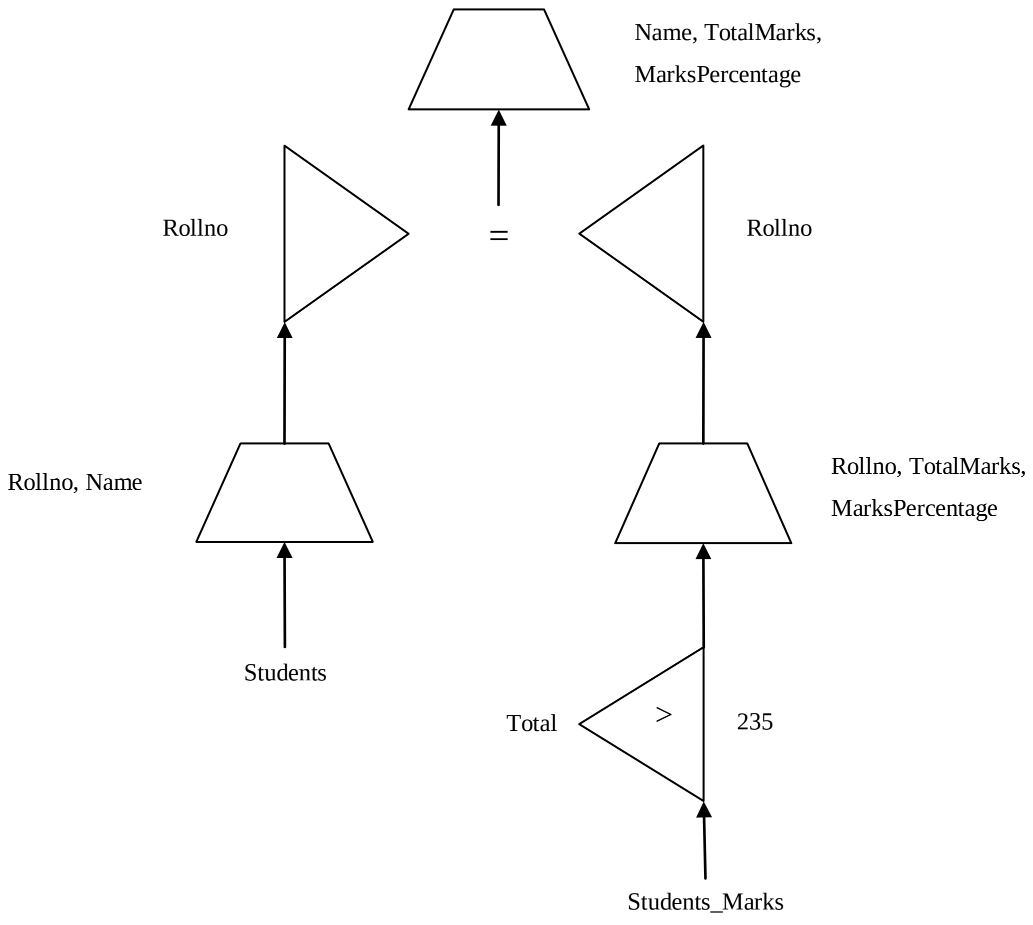

Figure 1. Query Graph for query in Example 1

3.2 Optimization using Tableaus

Tableaus notations for a subset of relational calculus queries are characterized by containing only AND-connected terms and no universal quantifiers [2]. Thus tableau queries are a particular kind of conjunctive queries. Tableaus are specialized matrices, the columns of which correspond to the attributes of the underlying database schema. The first row of the matrix serves the same purpose as the target list of a relational calculus expression. The other rows describe the predicate.
Fig 2 illustrates the construction of a tableau representing the query of Example 1. It starts with tableaus for single relations and proceeds by combining these tableaus into new tableaus for larger and larger sub expressions. Distinguished variables are denoted by a's; non-distinguished ones are denoted by b's. Sets of tableaus are used for representing general conjunctive queries [4]. The tableau building starts with creating a temporary table having only those attributes which are required to solve the query. In this example the attributes required are Rollno, Name, Total and Percentage. Next the tables in query are processed and the corresponding values from the tables are entered into this temporary table created turn by turn. Next the values are compared from the where clause and the values that are needed to be compared are changed. The marks values are changed and the values that satisfy the condition are kept and rest all are removed. Next the Rollnos from both the tables are compared and the values are combined in one table. Once they are combined, the values for rollno are changed from a to b. after

the values are changed to b, the values to be projected are shown to the user.

T (Students) =

Rollno	Name	Total	Percentage
a_1	a_2		
a_1	a_2		
a_1	a_2		
a_1	a_2		
a_1	a_2		

T (Marks) =

Rollno	Name	Total	Percentage
a_1		a_3	a_4
a_1		a_3	a_4
a_1		a_3	a_4
a_1		a_3	a_4
a_1		a_3	a_4

T ($\sigma_{\text{Total>235}}$) =

Rollno	Name	Total	Percentage
a_1		239	a_4
a_1		274	a_4
a_1		236	a_4

T (Join (Students.Rollno = Marks.Rollno) $\sigma_{\text{Total>235}}$) =

Rollno	Name	Total	Percentage
a_1 a_1	a_2	239	a_4
a_1 a_1	a_2	274	a_4
a_1 a_1	a_2	236	a_4

T (Project (Name, Total, Percentage) =

Rollno	Name	Total	Percentage
b_1 b_1	a_2	239	a_4
b_1 b_1	a_2	274	a_4
b_1 b_1	a_2	236	a_4

Figure 2. Tableaus for Example 1

3.3 Optimization of Queries having Aggregates

The aggregates in the query are the statements like group-by, having, min, max, etc. These functions complicate the query processing a bit. Thus the optimization of queries having aggregates needs a proper way. The optimization of queries with aggregates is considered less. Also the optimizers do not flatten the views in queries that reference views with aggregates [5].

To optimize queries with aggregate two steps are required, namely

- Transformations
- Optimization algorithms

Transformation is needed to optimize the queries containing group-by and join and optimization algorithms are required to incorporate the transformation [3]. In this process efforts are made such that the search space should not increase dramatically.

3.3.1. Transformations for Single Block SQL

The single block SQL are the SQL queries having no subquery. Transformations for making group-by to precede join is possible under many conditions which fall in following categories:

1. Conditions that depend on the schema and nature of join predicate and independent of nature of the aggregating functions.
2. Conditions that depend on specific properties of aggregating functions which leads to introduction of additional group-by nodes.
   a) Conditions that do not require introducing derived columns
   b) Conditions that require use of derived columns

Category 1 applies to aggregating functions. There are different ways of evaluating group-by and join predicates as shown in Fig 3.
Early group-by reduces the input size of join. Thus tree1 can be transformed to tree2. Tree3 represents the stagewise group-by from properties of aggregate functions which lead to introduction of group-by nodes. These aggregate functions follow property –
Agg (S U S') = Agg (Agg (S) U Agg(S'))

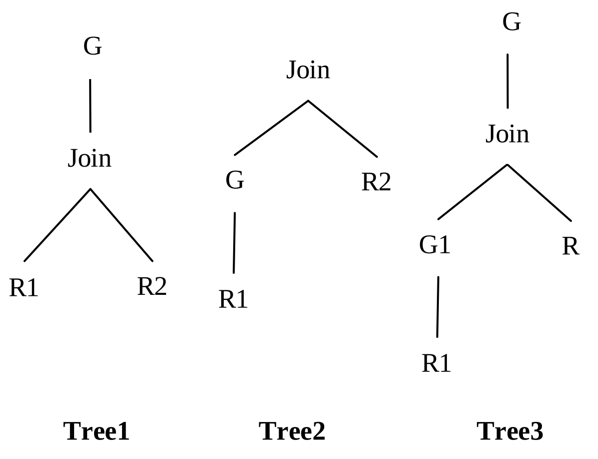

Figure 3. Single Block Query Transformations

3.3.2. Optimization Algorithm for Single Block Query

The optimizer decides when to apply transformation and in what form. The optimization in presence of group-by and aggregate functions has few problems which can be stated as:

- Increase in the number of operators.
- Effect on the physical properties.

The transformations sometime add some operators to original query. With these there are physical changes also which can cause increase in the width of intermediate relations or change in order.
The transformation of group-by introduces additional group-by operators which increases the space many times. Further, it is not possible to compare two segments of plan locally. As one plan may be efficient according to current query but other plan may reduce the cost of future joins. Another problem is that if there are more than one aggregation functions on the same column, then the width of the relation after application of group-bys will increase.

Greedy Conservative Heuristic

Greedy Conservative Heuristic is a technique to optimize the single block queries with group-by by exploiting the transformations [6]. It ensures that the quality of the plan is similar to the execution plans generated by the optimizer and there is no overhead in optimization.
This technique places group-by preceding the join if and only if the following conditions hold:

1. It is semantically correct.
2. The record width in the output of group-by is no longer than the one in its input.
3. It results in a cheaper plan for that join.

It considers only a single ordering of group-by columns for each join. Thus when sort-merge is used, a major to minor ordering is choose which is same for the join node. It ensures that the plan has high quality for which it chooses such a plan in which the size of intermediate results does not increase. Greedy Conservative Heuristic increases the quality of plan with modest increase in the optimization overhead.

4. CONCLUSION

Different techniques using different representations show that there are many other ways to represent query other than the query trees. The query graph is simple method to represent the query. Tableaus on the other hand are simple to define, use and implement. They not only provide a way of representing the query but also define a method of query optimization in an efficient way. Graphs and tableaus provide a very easy way to optimize the query in databases.
Optimization of query blocks is quiet complicated and requires a different method for optimization of single block queries. The multi block queries are yet more difficult to optimize and require much of the processing. The greedy method used for optimization of single block queries can be extended to optimize the multi block queries.
Query optimization still is a field where more work can be done in distributed and deductive databases.

5. REFERENCES

[1] Abdullah Dilsat : Query Optimization in Distributed Databases. *Report, Middle East Technical University,* December 2003.

[2] Aho, A.V., Sagiv,Y. and J. D. Ullman: Efficient optimization of a class of relational expressions. *ACM Trans. Database Systems.* 4, 4, p- 435-454, 1979.

[3] Chaudhuri S. and K. Shim. Query optimization with aggregate views. *In Proceedings of the 5th International Conference on Extending Database Technology*, Avignon, France, March 1996.

[4] Chaudhuri S. and K. Shim: An Overview of Cost-based Optimization of Queries with Aggregates. *IEEE DE Bulletin*, Sep. 1995. (Special Issue on Query Processing).

[5] Chaudhuri S. and K. Shim: Including group-by in query optimization. *In Proceedings of the 20th International VLDB Conference*, Santiago, Chile, Sept 1994.

[6] Chaudhuri S.: An Overview of Query Optimization in Relational Systems ; *Pods'09, ACM New York, NY, USA,* Year 1998.

[7] Leee Chiang, Chih Chi-Sheng and Chen Yaw-Huei : Optimizing large join queries using a graph-based approach. *IEEE Trans on Knowledge and Data Eng.*2001,13(2): p-298-315, 2001.

[8] Matthias Jarke, Jurgen Koch: Query Optimization in Database Systems. *ACM Computing Surveys*, Vol. 16, Issue 2, 1984.

[9] Sagiv, Y.: Optimization of Queries in Relational Databases. *UMI Research Press*, Ann Arbor, Michigan, 1981.

[10] Sukheja Deepak and Umesh Kumar Singh : A Novel Approach of Query Optimization for Distributed Database Systems. *IJCSI International Journal of Computer Science Issues*, Vol. 8, Issue 4, No 1, July 2011.

[11] Yao S.B.: Optimization of query evaluation algorithms. *ACM Trans. Database Syst.* Vol 4, 2 (June), p-133-155, 1979.

[12] Youssefi, K. and E. Wong, : Query processing in a relational database management system. *In Proceedings of the 5th International Conference on Very Large Data Bases* (Rio de Janeiro, Oct. 3-5). IEEE, New York, 1979, pp. 409-417.