A Survey on Query Processing and Optimization in Relational Database Management System

Saurabh gupta
Computer Science and Engineering Department, OPJindal Institute of Technology, Raigarh

Gopal Singh Tandel
Computer Science and Engineering Department, OPJindal Institute of Technology, Raigarh

Umashankar Pandey
Computer Science and Engineering Department, OPJindal Institute of Technology, Raigarh

Abstract - The query processer and optimizer is an important component in today's relational database management system. This component is responsible for translating a user query, usually written in a non-procedural language like SQL – into an efficient query evaluation program that can be executed against database. In this paper, we identify many of the common issues, themes, and approaches that extend this work and the settings in which each piece of work is most appropriate. Our goal with this paper is to be a "value-add" over the existing papers on the material, providing not only a brief overview of each technique, but also a basic framework for understating the field of query processing and optimization in general.

Keywords: Query processor, Query optimizer, RDBMS, Relational query language, SQL

I. INTRODUCTION

Relational query languages provide a high level "declarative" interface to access data stored in relational database. [3]. One of the fundamental breakthroughs of Codd's relational data model was the identification of how to take declarative, logic-based formulation of a query and convert it into an algebraic query evaluation tree. This enabled many physical data independence and promised many benefits: the database administrator and the DBMS optimizer became free to choose among many different storage formats and execution plans to answer a declarative query. The challenge, since then, has been how to deliver on these promises – regardless of where or how the data is laid out, how complex the query is , and how unpredictable the operating environment is.[2]

1.1 Query processing in relational algebra

The conventional method of processing a query in a relational DBMS is to parse the SQL statement and produce a relational calculus-like logical representation of the query, and then to invoke the query optimizer, which generates a query plan. The plan is fed into an execution engine that directly executes it, typically with little or no runtime decision – making (Figure1.1) [2]

The query plan can be thought of as a tree of unary and binary relational algebra operators, where each operator is annotated with specific details about the algorithms to use (e.g. nested loops join versus hash join) and how to allocate resources (e.g. memory). In many cases the query plan also includes low – level "physical" operators like sorting, network shipping, etc. that do not affect the logical representation of data.[2]

The query processing is divided into three major stages:

First stage is statistics generation that is done offline (typically using the RUNSTATS or UPDATE STATISTICS command) on the tables in the database. [2]

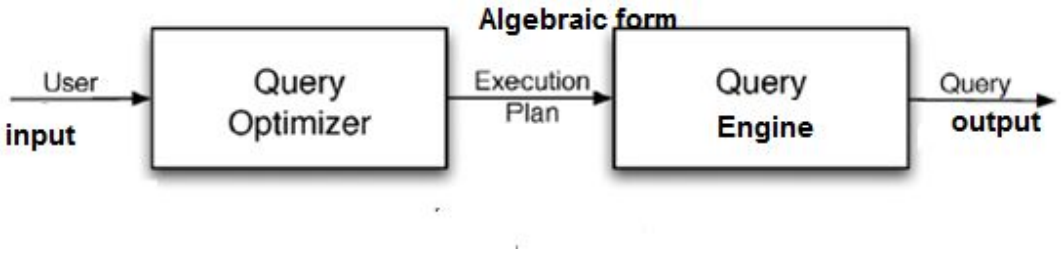

Figure 1.1 queries processing in database

The second stage, which is normally done at runtime, is query optimization. Optimization uses the combination of cost estimation, where the running times of query sub- expression are estimated. [2]
The final stage, query execution, is handled by an engine analogous to a virtual machine or interpreter for the complied query plan. There are several important aspects of query execution that are of note. [2]

1.2 Road Map

We begin with a brief introduction to query optimization in relational database system. We conclude the survey with a discussion of query processing and query optimization techniques.

1.3 Related Work

A number of surveys on query processing are related to this paper. We assume basic familiarities with many of the ideas of Graefe's survey on query execution techniques. [2]

II. BACKGROUND

2.1 Conventional optimization technique

We review some of the key concepts in single pass, non adaptive query optimization in relational database. General query optimization may consider Group By and multi-block SQL queries, the heart of cost – based optimization lies in selection ordering and join enumeration. [2]

2.1.1. Selection ordering

It refers to the problem of determining the order in which to apply a given set of commutative selection predicates (filter) to all the tuples of a relation, so as to find the tuples that satisfy all the predicates. Figure2.1 shows an example selection query over a persons relation and several query plans that query.

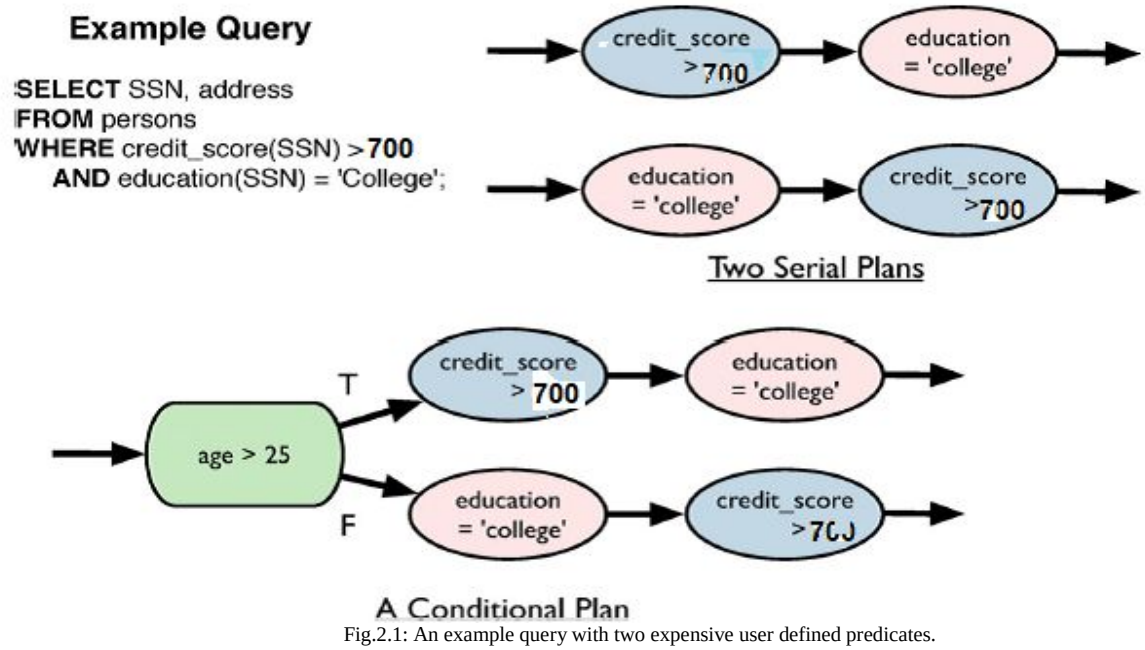

Fig.2.1: An example query with two expensive user defined predicates.

2.1.2. Multi-way Join queries

A select-project-join query, an optimizer has to make many choices, the most important being: access methods, join order, join algorithms, and pipeling. [2]
Access Method: The optimizer needs to pick an access method for each table in the query. Typically there are many choices, including a direct table scan, a scan over an index, an index based lookup on some predicate over that table, or an access from a materialized view.[2]
Join order: The access methods provide source tuples to the query plan. To join these tables, the optimizer, then chooses a join order.[2]
Pipelined plans: These plans execute all operators in the query plan in parallel, by sending output tuple of an operator directly to the next operator in the pipeline.[2]
2.1.3 Robust query optimization

Assuming only one plan can be chosen and the required probability distributions can be obtained. An approach along similar lines was proposed by Babcock and Chaudhari, called robust query optimization.[2]
2.1.4 Parametric query optimization

An alternative to finding a single robust query plan is to find a small set of plans that are appropriate for different situation. It postpone certain planning decisions to runtime, and are especially attractive in scenarios where queries are compiled once and executed repeatedly.[2]
2.2 Beyond the conventional query optimization

2.2.1. Adaptive query processing

The goal of adaptive query processing is to find an execution plan and a schedule that are well-suited to runtime conditions; it does this by interleaving query execution with exploration or modification of the plan or scheduling space. It uses the different operators for its operations like symmetric hash join operators, Eddy operators. It has some query processing technique like selection ordering for single table. Adaptive Greedy algorithm is based on the Greedy algorithm which continually monitors the selectivities of the query predicates using a random sample over the recent past , and ensures that the order used by the query processor is the same as the one that would have been chosen by the Greedy algorithm.[2]

So far we have different query optimization techniques. We end our survey paper with the summary of all the optimization techniques that can be used in query processing with the topics of research issues, author name, problem and proposed solution, advantages and disadvantages.

Table1. Summary of Query Optimization Techniques in Relational Databases: Problems/Solutions/Practices/Techniques [4]

S.No.	Research Issue	Authors	Problem discussed and proposed solution	Advantages	Disadvantages
1	Sub-query optimization in Oracle	Bellamokanda *et al.* [8]	Avoiding self joins, multi-joins, multi table access and handling NULL values. Proposed sub-query coalescing, sub-query removal using windows functions and NULL-AWARE ANTI-JOIN transformation techniques.	Improved query execution time.	Simulation results are achieved using parallel CPUs and other high level hardware which may be affected if degree of parallelism becomes low.
2	Paging Query Optimization of Massive Data	Sun *et al*.[9]	Paging query in large scale databases. Proposed a sharable stored procedure at server side and applying Oracle techniques like memory management, indexes, distribution of data and its files.	The proposed solution has huge effect on performance by utilizing the server and client resources efficiently.	The paper mainly focused on Oracle user guides.
3	Autonomic Computing in SQL Server	Mateen *et al*.[10]	Development of an Automatic Database Management Systems (ADBMS) Introduced self-optimization, self-healing, self-protection, self-inspection and self-organization key characteristics for ADBMS.	Main advantages of the proposed solution are: less DBA interaction with the system, efficient use of system resources, recovery and protection.	No simulation is performed to show the autonomic nature of the system.
4	An overview of query optimization	Chaudhuri [11]	Selecting the best execution plan for a query by an optimizer. Discussed the fundamental requirements for search spaces, accurate cost estimation technique and optimizer.	Best execution plan out of many candidates plans.	No optimization technique is discussed pertaining to memory issues linked with optimization.

5	Query optimization in centralized, distributed systems	Hameurlain [12]	The optimization in uni-processor, distributed and parallel processing environments. Discussed enumerative and random techniques, dynamic optimization algorithm, decentralized approach and adaptive query processing.	Enhance optimization of query in all the environments	The paper lacks algorithmic or prototype support.
6	Query optimization strategies for distributed databases	Lin [13]	Query optimization in distributed databases. The local data dictionary is added with sentence table to	These strategies enhance the query efficiency and reduce data transmission	As size of the corresponding table becomes large, it takes more CPU time and
			store mostly-used results to avoid the transmission of large data.		memory.
7	Optimizing Join queries	Zafarani *et al.*[14]	Join ordering in heterogeneous distributed databases.	Reduced calculation of join orders result in better response.	The simulation results shown in the paper are complex.
8	Query optimizers	Chaudhuri [15]	Cardinality estimation and effective cost estimation.	Best response time for a query.	No interface for the optimizer is discussed.
9	Query optimization strategy of the VDSI system database	Sun *et al.* [16]	Query optimization of large scale VDSI systems. Proposed indexes algorithm.	Reduces I/O operations and improves the response time of the query.	No cache technique is discussed.
10	Empirical evaluation of *LIKE* operator in Oracle	Gupta *et al.* [17]	Performance issues with *LIKE* operator, DB design and indexes	High throughput and best response time is achieved.	The paper only discusses *LIKE* operator, but the experimental diagrams and results are about indexes.
11	Query Optimization Techniques for Partitioned Tables	Herodotou *et al.[18]*	Optimization of SQL queries which are running over partitioned tables. Proposed partitioned aware technique for the optimizer.	Fast query processing, access of data in parallel fashion, efficient mechanism to load data and to maintain statistics.	Ambiguity about the type of partition for which the proposed technique can perform better.
12	Query processing and optimization in Oracle RDB	Antoshenkov *et al.[19]*	Query processing and its optimization in Oracle relational databases by reworking the query optimizer.	Improved compilation time to reduce complexity and to handle large amount of data.	Partitioning can be introduced while dealing with large amount of data.
13	Multi-Table Joins Through Bitmapped Join Indices	O'Neil *et al.* [20]	Query performance in decision support systems and OLAP environments by introducing a method to execute the common multi-table joins (Star join).	Better evaluation plans.	Indexes seriously affect performance when DML operations are performed.

14	Power Hints for Query Optimization	Bruno *et al.* [21]	To improve the poor plan selected by optimizer by query hints. store mostly-used results to avoid the transmission of large data.	Better query plan.	Query hints are a non-trivial complex method to achieve better plan. memory.

III. CONCLUSION

In this paper, we have made an attempt to present a detailed review of query optimization techniques. It is hard to capture the breadth and depth of this large body of the work in short article. The main idea behind this research is to review various techniques to implement query processing and optimization in an effective manner. Though the traditional, single-pass, optimize-then-execute strategy for query execution has served the database community quite well since the 1970. As queries have become more complex and widespread, however, they have started to run into limitations. Query optimization techniques and approaches primarily focus centralized and distributed databases. The paper also highlighted merits of these techniques by critically analyzing them with respect to their utility and efficacy. Schemes for robust optimization, parametric optimization, and inter-query adaptivity alleviate some of these difficulties by reducing sensitivity to errors. A significant virtue of these methods is that they impose little runtime overhead on query execution perhaps even more importantly; they serve as a simple upgrade path for conventional single pass query processor.

Selection ordering is a much simpler problem than optimization complex wulti-way join queries, and this simplicity not only enables design of efficient algorithms for solving this problem.

. We have discussed the existing techniques and their implementation for the sake of optimizing query. We have identified some of the proposed techniques that lead towards achieving the key benefits of an optimized query as compare to an un-optimized query in terms of its throughput and response time.

REFERENCES

[1] J.C.Freytag, "The basic principle of query optimization in relational database management system", Europena computer- industry research center GmbH, Arabellaster.17, D-8000 Muenchen 81 west Germany, Internal report IR-KB-59,20-March-1989.

[2] A.Despande, Z.Ives and V.Raman,"Adaptive query processing", Foundation and trend in databases, vol. 1, No. 1(2007) 1-140

[3] S.Chaudhari,"An overview of query optimization in Relational System", Microsoft research, One Microsoft Way, Redmond,WA 98052

[4] M.Khan, M.N.A.Khan, "Exploring query optimization techniques in Relational Databases", *SZABIST, Islamabad, Pakistan,* International Journal of Database Theory and Application Vol. 6,No.3,june,2013

[5] Y. Ioannidis, "Query Optimization", Journal ACM Computing Surveys (CSUR), (1996).

[6] D. Li, L. Han and Y. Ding, "SQL Query Optimization Methods of Relation Database System", Computer Engineering and Applications (ICCEA), (2010).

[7] S.Chande and M. Sinha, "Genetic optimization for the join ordering problem of database Queries", India Conference (INDICON), (2011).

[8] S.Bellamokanda, R. Ahmand and A. Witkowski, "Enhanced Subquery Optimizations in oracle", Proceeding of the VLDB Endowment, (2009).

[9] F. Sun and L. Wing, "Paging Query Optimization of Massive Data in Oracle 10g Database", Computer and Information Science and Service System (CSSS), IEEE International Conference, (2011).

[10] A. Mateen, B. Raza and T. Hussain, "Autonomic Computing in SQL Server", Computer and Information Science, Seventh IEEE/ACIS International Conference, (2008).

[11] S. Chaudhri, "An overview of query optimization in relational systems", Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems**,** (1998).

[12] A. Hameurlain, "Evolution of Query Optimization Methods: From Centralized Database Systems to Data Grid Systems", Proceedings of the 20th International Conference on Database and Expert Systems Applications, (2009).

[13] X. Lin, "Query Optimization Strategies and Implementation Based on Distributed Database", ComputerScience and Information Technology, 2nd IEEE International conference, (2009).

[14] E. Zafarani, M. Reza, H. Asil and A. Asil, "Presenting a New Method for Optimizing Join Queries Processing in Heterogeneous Distributed Databases", In Knowledge Discovery and Data Mining,WKDD '10, (2010).

[15] S. Chaudhuri, "Query optimizers: time to rethink the contract?", Proceedings of the 35th SIGMOD international conference on Management of data, (2009).

[16] P. Sun, Z. Zhao and Z. Ge, "The research on the query optimization strategy of the VDSI system database", Computational Intelligence and Industrial Applications, PACIIA 2009, Asica Pacific Conference, (2009).

[17] M. Gupta and P. Chandra, "An Empirical Evaluation of LIKE Operator in Oracle", BVICAM'S International Journal of Information Technology (BIJIT), (2011).

[18] H. Herodotou, N. Borisov and S. Babu, "Query Optimization Techniques for Partitioned Tables", ACM SIGMOD International Conference on Management of data, (2011).

[19] G. Antoshenkov and M. Ziauddin, “Query processing and optimization in Oracle Rdb”, The VLDB Journal The International Journal on Very Large Data Bases, (1996).
[20] P. O’Neil and G. Graefe, “Multi-Table Joins Through Bitmapped Join Indices”, ACM SIGMOD, (1995).
[21] N. Bruno, S. Chaudhuri and R. Ramamurthy, “Power Hints for Query Optimization”, IEEE International Conference on Data Engineering, (2009).