Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 8, August 2016, pg.148 – 154

An Extensive Survey on Various Query Optimization Techniques

A. Regita Thangam[1], Dr. S. John Peter[2]

[1]Research Scholar, Department of Computer Science, St.Xavier's College, Palayamkottai, Tamil Nadu, India
[2]Associate Professor and Head, Department of Computer Science, St.Xavier's College, Palayamkottai, Tamil Nadu, India
[1] regitaraja@gmail.com, [2] jaypeeyes@rediffmail.com

Abstract*— As the database management field has diversified to consider settings in which queries are increasingly complex, statistics are less available, or data is stored remotely, there has been an acknowledgment that the available optimization techniques are insufficient. This has led to a plethora of new techniques, generally placed under the common banner of optimizing complex queries that focus on rewriting the complex queries in a simple manner. Query optimization is the bottleneck of database application performance especially those which store history i.e. data warehouse. SQL is used as query language because most data warehouses are based on relational or extended relational database system. In this survey paper, we identify many of the common issues, themes, and approaches that pervade 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 understanding the field of query processing in general and also to reduce the complexity of the queries to enhance the query processing and optimization engines.

***Keywords*— Data warehouse, query optimization, query processing, complex queries, SQL.**

I. Introduction

The query optimizer is the component of a database management system to optimize the query. It is used to select an efficient execution strategy for processing a query. Query optimization is a function of many relational database management systems in which multiple query plans for satisfying the queries are examined and a good query plan is identified to minimize the use of certain resources like I/O by selecting the best query access plan.

In present scenario data warehouses & mining turned out to be the common basis for the integration and analysis of data in modern enterprises. The goal of a data warehouse is to provide analysts and managers with strategic information about the key figures of the underlying business. Since micro data are of no interest at this level, almost all queries on data warehouses involve aggregates. In large data warehouse systems, it is critical to optimize query workloads to maximize system utilization and minimize processing times.

The database administrator and the Data Base Management System 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.

II. QUERY PROCESSING AND OPTIMIZATION

With the growing number of large data warehouses for decision support applications, efficiently executing aggregate queries is becoming increasingly important. Aggregate queries are frequent in decision support applications, where large history tables often are joined with other tables and aggregated. Because the tables are large, better optimization of aggregate queries has the potential to result in huge performance gains. Unfortunately, aggregation operators behave differently from standard relational set operators like Select, Project, and Join. Thus, existing rewrite rules for optimizing queries almost never involve aggregation operators. To reduce the cost of executing aggregate queries in a data warehousing environment, frequently used aggregates are often precomputed and materialized. These materialized aggregate views are commonly referred to as summary tables. Summary tables can be used to help answer aggregate queries other than the view they represent, potentially resulting in huge performance gains. However, no efficient algorithms exist for replacing base relations in aggregate queries with summary tables so the full potential of using summary tables to help answer aggregate queries has not been realized.

III.VARIOUS APPROACHES FOR QUERY OPTIMIZATION

3.1 Optimization in Relational databases

A. K. Chandra and P. M. Merlin [1] defined the class of conjunctive queries in relational data bases, and the generalized join operator on relations. The generalized join plays an important part in answering conjunctive queries, and it can be implemented using matrix multiplication. This allows the possibility of finding smaller queries equivalent to a given one without having to obtain the minimal query. The minimal query can, however, be used to determine a program whose running time is within a constant of optimal for every data base.

Chen M. S., Yu P. S., and Wu K. L [10] focussed on two major issues to optimize the execution of multi-join queries. They were scheduling the execution sequence of multiple joins within a query, and determining the number of processors to be allocated for the execution of each join operation. Note that different join execution sequences for a query will result in different execution costs [2]. Also, the execution time of a join in a multiprocessor system strongly depends on the number of processors allotted for the execution of that join [5]. Finally the two-step approach proposed in [10], which first applies the join sequence heuristic to build a bushy tree as if under a single processor system, and then, in light of the concept of synchronous execution time, allocates processors to execute each join in the bushy tree in a top-down manner, emerges as the best solution to minimize the query execution time.

Rank join operators combine objects of two or more relations and output the k- combinations with the highest aggregate score. The rank join problem has been dealt with in the literature [23] by extending rank aggregation algorithms [21] to the case of join in the setting of relational databases. Davide Martinenghi and Marco Tagliasacchi [29] proposed the Cost-Aware with Random and Sorted access (CARS) pulling strategy for retrieving the k- combinations with the highest aggregate score that can be formed by joining the results of heterogeneous search engines. They optimized such a strategy with respect to an additive cost model that considers both sorted access and random access. They validated the proposed strategy with experiments on both real and synthetic data sets. It showed that CARS outperforms prior proposals and that its overall access cost is always within a very short margin from that of an oracle-based optimal strategy.

Christian Politz et al. [31] explained the problem of ranking under budgets on loading and computational costs and introduced a budget-aware learning to rank approach that limits the cost for evaluating a ranking model, with a focus on very tight budgets that do not allow to fully evaluating at least for one time all documents for each term. The dependencies were used to estimate the use we gain when using postings of certain terms and applying specific rankers. The evaluation of our proposed solution of the optimization task showed better results compared with state-of-the-art budget aware ranking methods.

The efficient integration of preference querying into standard database technology is an important issue. Bernd Hafenrichter, Werner Kießling[24] presented a novel approach to relational preference query optimization based on algebraic transformations. The preference queries can be evaluated by preference relational algebra, extending classical relational algebra by two new preference operators. They have provided a series of novel transformation laws for preference relational algebra that are the key to algebraic optimization. This method could achieve excellent performance.

Mingsheng Hong et al. [27] proposed a Rule-based Multi-Query Optimization framework, called RUMOR, which naturally extends the rule-based query optimization and query-plan-based processing model used by the current RDBMS and stream systems. RUMOR provided a modular and extensible framework, enabling new optimization techniques to be developed and incorporated incrementally into the system. It also integrated the new and existing Multi-Query Optimization techniques for relational stream engines and for event engines. It is inspired by the classical Query Graph Model (QGM) of RDBMS [6], where query optimization techniques for single queries can be naturally modelled as transformation rules on query plans. The importance of Multi-Query Optimization, first studied in the context of relational database query processing [4], is recognized in J.Chen et al.

[17]. Hammad et al. [22] developed techniques to share work among multiple stream join operators which read the same input streams and have the same join predicate but potentially different window specifications.

Early aggregation is a technique for speeding up the processing of GROUP BY queries by reducing the amount of intermediate data transferred between main memory and disk. Larso [12] described the early aggregation into six algorithms. Three of the algorithms, namely sorting with replacement selection and the two algorithms based on hash partitioning, achieve much higher data reduction than the other three. All six algorithms benefit from early aggregation with grouping by hash partitioning producing the least amount of intermediate data. If the group size distribution is skewed, the overall reduction can be very significant, even with a modest amount of additional main memory.

With the recent interest in decision support systems and data warehousing has come to a demand for techniques to evaluate and optimize very complex relational queries involving both aggregation and joins. D.Chatziantoniou and K.A.Ross[14] define and examine a particular class of queries called group queries. The main characteristic of a group query is that it can be executed in a group by group fashion. They have provided a syntactic criterion for a query to be a group query. They provided two methods to identify group query components within such a query. They have provided an execution plan for evaluating group queries that partitions the relations, applies a simpler query to each partition, and unions the results. Because the query applied to each partition has simpler join conditions and grouping attributes, it is often easier to optimize. They have also presented some performance results using a commercial database system that indicate that large potential performance improvements are possible.

Complicated queries and views are hard to understand and maintain. Damianos Chatziantoniou and K. A. Ross [11] considered the queries involving selection, grouping and aggregation over the same groups, and proposed an extension of SQL syntax that allows the succinct representation of these queries. They proposed a new relational algebra operator Φ that represents several levels of aggregation over the same groups in an operand relation. Their queries were significantly shorter and simpler than their standard SQL counterparts, and were much more easily optimized. The syntax that they proposed was significantly more general than previous proposals, and enabled the concise solution of a number of well-known query representation problems.

zsoyo_glu et al. [3] extended relational algebra and relational calculus database query languages to manipulate set-valued attributes and to utilize aggregate functions. So they introduced three new operators, namely, pack, unpack and aggregation-by-template. They defined the set of extended relational algebra expressions containing aggregate functions and set-valued attributes. They also extended the relational calculus by allowing set-valued terms and variables, introducing type 2 range formulas and defining target alphas that provide the capability to output tuples with set valued components.

Ralf Rantzaua et al. [19] presented a classification of input data for algorithms that evaluate division within queries. In relational databases, universal quantification is implemented by the division operator (represented by ÷) of the relational algebra. They gave an overview of known and new algorithms to solve the universal quantification problem and to identify optimal algorithms for the division operator, for all possible inputs. They proposed a new operator, called set containment division that realizes set containment joins for data in first normal form.

N. Mamoulis[20] proposed the join of two relations on their set-valued attributes. He considered the various join types, namely the set containment, set equality and set overlap joins. He described existing signature-based algorithms for the set containment join and presents new techniques that employ inverted files. The signature-based methods are only appropriate for set equality joins. For the other join types, a version of Block Nested Loops algorithm was the most suitable algorithm. The set containment join can also be evaluated by joining the two inverted files. On the other hand, a method that joins two inverted files was found inappropriate for all join types.

P.E.O'Neil and G.Graefe [8] presented a method which uses join indices with compressed bitmap representations, which allow predicates restricting columns of descriptive tables to determine an answer set in the central detail table. This method uses different predicates on different descriptive tables in combination to restrict the detail table through compressed bitmap representations of join indices, and easily completes the join of the fully restricted detail table rows back to the descriptive tables. It is well known that the list of rows associated with a given index key value can be represented by a bitmap or bit vector. In a bitmap representation, each row in a table is associated with a bit in a long string, an N-bit string if there are N rows in the table, and the bit is set to 1 in the bitmap if the associated row is contained in the list represented; otherwise the bit is set to 0. The point of using bitmap indices, of course, is the tremendous performance advantage to be gained.

Patrick O'Neil, Dallan Quass[13] introduced two indexing structures, which we call Bit-Sliced indexes and Projection indexes. Indexes like these were used previously in commercial systems, Sybase IQ and MODEL 204, but never examined in print. As a new contribution, they had shown how ad-hoc Online Analytical Processing (OLAP) style queries involving aggregation and grouping can be efficiently evaluated using indexing and clustering, and they have introduced a new index type, Groupset indexes, that are especially well-suited for evaluating this type of query.

Hui Zhao et al. [30] presented on a global index based optimization strategy for range query and analysis, and do some tests to evaluate the correctness and efficiency in the end, the strategy is first checking whether user requests can be optimized by using the global index knowledge. If it does we use a customized tasks division method which is based on global index structure and query condition, result in reducing task number and input data, through which we not only gain better response time, but also optimize resource usage and task scheduling overhead. Finally, we experimentally verified the feasibility of this technology.

3.2 Optimization in distributed databases

The distributed query optimization is one of the hardest problems in the database area [25]. The optimizer, for example, determines which indices should be used to execute a query and in which order the operations of a query (e. g. joins, selects, and projects) should be executed. For a given SQL query, there is more than one possible algebraic queries. Some of these algebraic queries are better than others. The quality of an algebraic query is defined in terms of expected performance. Therefore, in the query optimization step that transforms the initial algebraic query using relational algebra transformations into other algebraic queries until the best one is found.

Swati Jain et al. [32] explained the major optimization issues being addressed in distributed databases. They explored the major principles of query optimization process with volcano query optimization. In order to examine the role of query optimization process in RDBMS, they proposed both static and dynamic process of optimization as well as all the general aspects of query optimization. But these strategies do not guarantee to find the optimal plan.

Modern database systems use a query optimizer to identify the most efficient plan to execute declarative SQL queries. The role of query optimizers is critical for the decision-support queries featured in data warehousing and data mining applications. Pawan Meena et al. [28] presented an abstraction of the architecture of a query optimizer and the technical constraints of advanced issues in query optimization.

Giovanni Maria Sacco[26] presented new approach to query optimization, Truly Adaptive Optimization(TAO). TAO is a unifying framework for query optimization problems and is composed of three elements, a fast solution space search algorithm, derived from A* algorithm, a relaxation technique which allows to specify a tolerance on the quality of the resulting query execution plan, a paradigm to prove the sub optimality of search subspaces. This method was applied to query optimization for databases distributed over a broadcast network and provided better performance.

In many applications, there is a need to integrate data and operations that are external to the database. For many applications, only part of the data that is needed may be stored in the database and much of the data may reside externally. Access to such external data is provided by a set of interface routines. Surajit Chaudhuri and Kyuseok Shim[16] described a comprehensive approach for optimization in the presence of foreign functions. They provided a declarative rewrite rule system which can be used to express semantics of foreign functions and also provided an algorithm to enumerate the equivalent queries. Their framework includes extensions to the cost model and query processing techniques that are necessary for foreign functions and it is used to optimize the relational queries where the database stores materialized views.

3.3 Optimization in data warehousing environment

A data warehouse is an electronic storage of huge amounts of data. It is also a system for retrieving and managing a data. It integrates data from different sources and provides the integrated data for subject-oriented purposes. Many methodologies had been proposed to tackle the problems of query processing in Data warehouse. M.C. Wu and A.P. Buchmann [15] proposed an encoded bitmap indexing, an extension of known bitmap indexing, for improving query processing in the data warehousing environment. They gave comparative performance analysis of simple and encoded bitmap indexes and the result showed that encoded bitmap indexes perform in most cases better.

Answering queries using materialized views was studied in [9]. However, this work had not considered queries that use aggregation. Ashish Gupta et al. [7] presented a powerful query rewrite rules for aggregation operators. Utilizing this rewrite rules, they gave an algorithm for answering aggregate queries using materialized aggregate views. This algorithm was very useful for decision support applications in data warehousing environments.

Albrecht et al. [18] proposed an extension of traditional query rewrite techniques. Derivability of multidimensional aggregates is the condition that has to be fulfilled to compute the result of an aggregate query based on the values of one or more aggregate views. They presented the conditions for derivability in a large number of relevant cases which go beyond previous approaches.

Ying Wah Teh et al. [33] introduced the various query processing techniques that are used in Data Warehousing queries. They compared the performance of the different query processing techniques and proposed a recommendation for Database Management Systems to select the most cost-effective query processing techniques based on the cost model.

IV. CONCLUSIONS

To a large extent, the success of a DBMS lies in the quality, functionality, and sophistication of its query optimizer, since that determines much of the system's performance. Query optimization is of great importance for the performance of a relational database, especially for the execution of complex SQL statements. A query optimizer determines the best strategy for performing each query. In this paper, we have carried out a decisive review of the most recent work carried out in this area. The strong points of various proposed techniques are summarised in Table I. So after reviewing different approaches it can be concluded that query optimization is a vast research area and can be extended in several ways. Despite many years of work, significant open problems remain. However, it is necessary for making effective contribution to the area of query optimization.

TABLE I
SUMMARY OF STRONG POINTS OF PROPOSED TECHNIQUES

Ref.No.	**Techniques Proposed**	**Strong Point**
1	Optimal Implementation of Conjunctive Queries in Relational Data Bases	To find the smaller query equivalent to the given conjunctive query whose running time is within a constant of optimal for every database.
3	Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregate Functions	Three new operators, namely, pack, unpack and aggregation-by-template were introduced to manipulate set-valued attributes and to utilize aggregate functions.
7	Aggregate-Query Processing in Data Warehousing	Query rewrite rules for aggregation operators were presented. Utilizing this rewrite rules, they gave an algorithm for answering aggregate queries using materialized aggregate views.
8	Multi-Table Joins Through Bitmapped Join Indices	Query performance in decision support systems and OLAP environments are improved by introducing a method to execute the common multi-table joins. The point of using bitmap indices is the tremendous performance advantage to be gained.
10	Optimization of Parallel Execution for Multi-Join Queries	They were scheduling the execution sequence of multiple joins within a query, and determining the number of processors to be allocated for the execution of each join operation.
11	Querying Multiple Features of Groups in Relational Databases	They proposed an extension of SQL syntax with a new relational algebra operator Φ that represents several levels of aggregation over the same groups in an operand relation. Their queries were significantly shorter and simpler than their standard SQL counterparts, and were much more easily optimized.
13	Improved Query Performance with Variant Indexes	They had shown how ad-hoc Online Analytical Processing (OLAP) style queries involving aggregation and grouping can be efficiently evaluated using indexing and clustering, and they have introduced a new index type, Groupset indexes, that are especially well-suited for evaluating these types of queries.
14	Groupwise Processing of Relational Queries	They defined a particular class of queries called group queries and also provided an execution plan for evaluating group queries that partitions the relations, applies a simpler query to each partition, and unions the results. Because the query applied to each partition has simpler join conditions and grouping attributes, it is often easier to optimize.
16	Query optimization in the presence of Foreign functions	They provided a declarative rewrite rule system which can be used to express semantics of foreign functions and also provided an algorithm to enumerate the equivalent queries.
19	Algorithms and Applications for Universal Quantification in Relational Databases	In relational databases, universal quantification is implemented by the division operator of the relational algebra. They gave an overview of known and new algorithms to solve the universal quantification problem and to identify optimal algorithms for the division operator, for all possible inputs.
20	Efficient Processing of Joins on Set-Valued Attributes	The join of two relations on their set-valued attributes was proposed. He considered the various join types, namely the set containment, set equality and set overlap joins and proposed efficient algorithms to evaluate these joins.
26	Truly Adaptive Optimization: The Basic Ideas	It is composed of three elements, a fast solution space search algorithm, derived from A* algorithm, a relaxation technique which allows to specify a tolerance on the quality of the resulting query execution plan, a paradigm to prove the sub optimality of search subspaces.

27	Rule-Based Multi-Query Optimization	It provided a modular and extensible framework, enabling new optimization techniques to be developed and incorporated incrementally into the system. It also integrated the new and existing Multi-Query Optimization techniques for relational stream engines and for event engines.
30	MapReduce model-based optimization of range queries	The global index based optimization strategy first checks whether the user requests can be optimized by using the global index knowledge. If it does it uses a customized tasks division method, result in reducing task number and input data, through which we not only gain better response time, but also optimize resource usage and task scheduling overhead.
32	Performance Analysis of Optimization Techniques for SQL Multi Query Expressions over Text Databases in RDBMS	In order to examine the role of query optimization process in RDBMS, they proposed both static and dynamic process of optimization as well as all the general aspects of query optimization. But these strategies do not guarantee to find the optimal plan.

REFERENCES

[1] A.K. Chandra and P.M. Merlin, "Optimal Implementation of Conjunctive Queries in Relational Data Bases," Published in the Proc. Ninth Ann. ACM Symp. Theory of Computing (STOC), 1977.

[2] P.G. Selinger, M.M. Astrahan, D.D. Chamberlin, R.A. Lorie, and T.G. Price, "Access Path Selection in a Relational Database Management System", Published in *Proc. ACM SIGMOD,* pp. 23-34, 1979.

[3] G. O. zsoyo_glu, Z.M.O.Zsoyo_glu, and V. Matos, "Extending Relational Algebra and Relational Calculus with Set-Valued Attributes and Aggregate Functions", Published in ACM Trans. Database Systems, vol. 12, no. 4, pp. 566-592, 1987.

[4] T. K. Sellis, "Multiple-query optimization", Published in ACM TODS,13(1):23–52, 1988.

[5] M. S. Lakshmi and E'. *S.* Yu, "Effectiveness of Parallel Joins", Published in *IEEE Trans. Knowledge and Data Eiig.,* vol. 2, no. 4, pp. 410424, Dec. 1990.

[6] H. Pirahesh, J. M. Hellerstein, and W. Hasan, "Extensible/rule based query rewrite optimization in starburst", Published in Proc.SIGMOD, pages 39–48, 1992.

[7] Ashish Gupta, Venky Harinarayan, Dallan Quass, "Aggregate-Query Processing in Data Warehousing", Published in the Proceedings of the 21st VLDB Conference Zurich, Swizerland, 1995.

[8] P. E. O'Neil and G. Graefe, "Multi-table joins through bitmapped join indices", Published in *SIGMOD Record*, 24(3):8–11, 1995.

[9] S. Chaudhuri et al, "Query Optimization in the presence of Materialized Views" Published in ICDE, 1995.

[10] Chen M. S., Yu P. S., and Wu K. L., "Optimization of Parallel Execution for Multi-Join Queries", Published in *IEEE Transaction on Knowledge and Data Engineering*, vol. 6, no. 3, 1996.

[11] Damianos Chatziantoniou, Kenneth A. ROSS, "Querying Multiple Features of Groups in Relational Databases", published in Proc. VLDB '96 of the 22th International conference on Very Large Data Bases, Pages 295-306.

[12] P.-A. Larso, "Grouping and Duplicate Elimination: Benefits of Early Aggregation," Published in Technical Report MSR-TR-97-36, Microsoft Research, 1997.

[13] Patrick O'Neil, Dallan Quass, "Improved Query Performance with Variant Indexes" Published in the Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 38-49, 1997.

[14] D. Chatziantoniou and K.A. Ross, "Groupwise Processing of Relational Queries", Published in the Proc. 23rd Int'l Conf. Very Large Databases (VLDB), pp. 476-485, 1997.

[15] M.C. Wu and A.P. Buchmann, "Encoded Bitmap Indexing for Data Warehouses", Published in Proc. 14th Int'l Conf. Data Eng. (ICDE), pp. 220-230, 1998.

[16] Surajit Chaudhuri, Kyuseok Shim, "Query optimization in the presence of Foreign functions", Published in the Proceedings of the 19th International Conference on Very Large Data Bases 03/2000;

[17] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ, "A scalable continuous query system for internet databases", Published in Proc. SIGMOD, pages 379–390, 2000.

[18] J. Albrecht, W. Hümmer, W. Lehner, L. Schlesinger, "Query Optimization By Using Derivability In a Data Warehouse Environment", Published in the Proceedings of the 3rd ACM international workshop on Data warehousing and OLAP, DOLAP -2000, pages 49 – 56.

[19] Ralf Rantzaua, Leonard D,. Shapirob, Bernhard Mitschanga and Quan Wangc, "Algorithms and Applications for Universal Quantification in Relational Databases", Published in Information Systems, Special issue: Best papers from EDBT 2002, Volume 28, Issue 1-2, 01 March 2003.

[20] N. Mamoulis, "Efficient Processing of Joins on Set-Valued Attributes," Published in the Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 157-168, 2003.

[21] R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware", Published in Computer and System Sciences, vol. 66, no. 4, pp. 614-656, 2003.

[22] M. A. Hammad, M. J. Franklin, W. G. Aref, and A. K.Elmagarmid, "Scheduling for shared window joins over datastreams", published in Proc. VLDB, pages 297–308, 2003.

[23] I.F. Ilyas, W.G. Aref, and A.K. Elmagarmid, "Supporting Top-k Join Queries in Relational Databases", Published in VLDB J., vol. 13, no. 3, pp. 207-221, 2004.

[24] Bernd Hafenrichter, Werner Kießling, "Optimization of Relational Preference Queries", published in Proc. ADC '05 Proceedings of the 16th Australasian database conference - Volume 39.

[25] Alaa Aljanaby, Emad Abuelrub, Jordan and Mohammed Odeh, "A Survey of Distributed Query Optimization", published in The International Arab Journal of Information Technology, Vol. 2, No. 1, January 2005.

[26] Giovanni Maria Sacco, "Truly Adaptive Optimization: The Basic Ideas", published in Database and Expert Systems Applications (DEXA), volume 4080 of Lecture Notes in Computer Science, page 751-760, Springer, 2006.

[27] Mingsheng Hong, Mirek Riedewald, Christoph Koch, Johannes Gehrke, Alan Demers, "Rule-Based Multi-Query Optimization", Published in Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT - March 24–26, 2009.

[28] Pawan Meena, Arun Jhapate & Parmalik Kumar, "Framework for Query Optimization", published in the International Journal of Computer Science and Information Security, Vol. 9, No. 10, October 2011.

[29] Davide Martinenghi and Marco Tagliasacchi, " Cost-Aware Rank Join with Random and Sorted Access", Published in the IEEE Transactions On Knowledge And Data Engineering, VOL. 24, NO. 12, DECEMBER 2012.

[30] Hui Zhao, Shuqiang Yang, Zhikun Chen, Songcang Jin, Hong Yin and Long Li, "MapReduce model-based optimization of range queries", Published in 2012, 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2012).

[31] Christian Politz and Ralf Schenkel," Ranking under tight budgets", Published in 2012 23rd International Workshop on Database and Expert Sytems Applications.

[32] Swati Jain and Paras Nath Barwal, "Performance Analysis of Optimization Techniques for SQL Multi Query Expressions over Text Databases in RDBMS", Published in the International Journal of Information & Computation Technology, Volume 4, no. 8, 2014.

[33] Ying Wah Teh, A. B. Zaitun, "Query Processing Techniques in Data Warehousing Using Cost Model", Published in the Electronic Journal of Information Systems in developing Countries, Volume 3, 2000.