International Journal of Computer Applications (0975 – 8887)
Volume 114 – No. 4, March 2015

A Review Paper on Different Approaches for Query Optimization using Schema Object base View

Dhaval Patel
Research Scholar, CSE Department
Parul Institute of Technolody,
Limda, Vadodara, India.

Pratik Patel
Asst. Prof., CSE Department
Parul Institute of Technolody,
Limda, Vadodara, India.

ABSTRACT

Mining of Data is the extraction of hidden prognosticative information from large databases or set of data, is a strong new technology with great prospective to help companies focus on the most important information in their data warehouses. Query optimization is a purpose of many relational database management systems. The query optimizer experiments to dictate the most efficient way to implement a given query by examining the possible query plans. In this paper, the different techniques is given for optimizing query using schema based and materialized views in data warehouse namely- view adaptation & synchronization, view selection and view maintenance.

Keywords

Query optimization, materialized view

1. INTRODUCTION

Predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions is tooled by Data mining[1]. The automated, potential inspects recommended by data mining move beyond the analyses of past occurrences provided by backward-looking tools typical of decision support systems. Massive quantities of data already are collected and refined by most companies. Data mining techniques can be implemented promptly on existing software and hardware platforms to strengthen the value of existing information resources, and can be integrated with new products and systems as they are brought on-line.

Query optimization is a consequence of many relational database administration organizations. Query Optimization is the procedure of choosing the most systematic technique to accomplish a SQL statement. When the cost-based optimizer was provided for the first time with Oracle7, Oracle supported only standard relational data[2]. The introduction of objects enlarged the maintained data types and functions. The aim is to attempt them all out, but it requires deciding in what order. What interchange of tastes will maximize the comprehensive fulfillment of palate? Although much less pleasurable and instinctive, that is the type of problem that query optimizers are called to interpret. Given a query, there are many programs that a database management system (DBMS) can track to procedure it and manufacture its answer. All programs are identical in terminology of their final output but different in their value, i.e., the amount of time that they require to pass.

2. QUERY FLOW

The first step in processing a query submitted to a DBMS is to convert the query into a form usable by the query processing engine. High- level query languages such as SQL represent a query as a string, or sequence, of characters. Certain sequences of characters represent different types of tokens such as keywords, operators, operands, literal strings, etc[1][2].

The primary job of the parser is to extract the tokens from the raw string of characters and translate them into the corresponding internal data elements for example relational algebra operations and operands and structures for example query tree, query graph. The last job of the parser is to check the validity and syntax of the actual query string[1].

In second stage, the query processor applies instructions to the internal data structures of the query to convert these structures into complement, but more efficient representations. The rules can be based upon mathematical models of the relational algebra expression and heuristics, upon cost approximates of different algorithms or techniques applied to operations or upon the semantics within the query and the relations it necessitates[2]. Selecting the absolute rules to apply, when to apply them and how they are applied is the function of the query optimization engine.

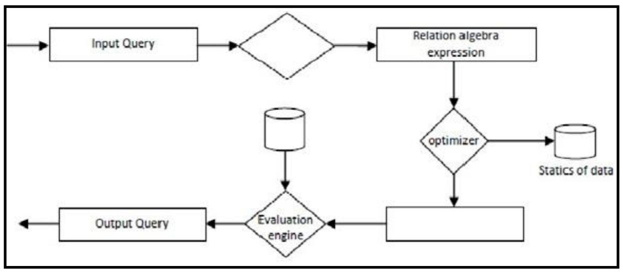

Figure 1.3: Steps in query processing

The final step in processing a query is the evaluation phase. The best estimation plan candidate generated by the optimization engine is chose and then accomplished. Besides processing a query in a simple sequential manner, some of a query's individual operations can be processed in parallel either as unconventional procedures or as interdependent pipelines of procedures or threads[2].

3. APPROACHES FOR QUERY OPTIMIZATION

3.1 An Efficient Processing of Queries with Joins and Aggregate Functions- View Selection

In this technique, it proposes a new method for processing the queries including both joins and aggregate functions. The method performs grouping dimension tables with group-by conditions at first and then processes joins by using bitmap join indices[3]. It is very important to procedure efficiently expensive queries including joins and/or aggregate functions in data warehousing environment since there resides extensive volume of data and the procedure of these queries takes much time. This allows to process aggregate functions by accessing fact tables only, thus it can reduce the serious performance degradation of existing methods. In order to show the superiority of the method, they develop a cost model for both the proposed and the existing ones, and perform extensive simulations based on the TPC-H benchmark.

They have proposed a technique that performs grouping of the dimension table in advance and removes the number of disk accesses in grouping the joined table. The dimension table involves the grouping attributes for aggregate functions and the fact table involves the aggregate attributes[3][4]. With the characteristics, grouping of the dimension table is performed by the proposed procedure which is quite smaller than the fact table. This approach is able to perform aggregate functions efficiently using the bitmap join indices developed for processing joins.

3.2 A Comprehensive Analysis of Materialized Views

Data in a warehouse can be perceived as a collection of materialized views that are generated as per the user requirements specified in the queries being generated against the information contained in the warehouse. User requirements and restraints frequently change over time, which may evolve data and view definitions stored in a data warehouse dynamically. The current requirements are changed and some novel and innovative requirements are added in order to deal with the latest business scenarios[5]. In fact, data conserved in a warehouse along with these materialized views must also be updated and maintained so that they can deal with the changes in data sources as well as the requirements stated by the users. Selection and maintenance of these views is one of the vital assignments in a data warehousing environment in order to provide optimal effectiveness by decreasing the query response time, query processing and maintenance costs as well.

This analysis has presented different approaches being proposed by different researchers to deal with the materialized views in data warehouse that is to say view adaptation & synchronization, view selection and view maintenance[6]. They have examined these techniques on various parameters and provided a comparative study in a tabular manner.

3.3 Algorithms for Materialized View Selection and Maintenance

The materialization of all views is practically impossible because of the materialized view storage space and maintenance cost constraint thus proper materialized views selection is one of the intelligent decisions in designing a data warehouse to get optimal efficiency. It presents a framework for selecting better materialized view so as to achieve the effective combination of good query response time, less query processing cost and low view maintenance cost in a specified storage space constraint[7]. Query frequency cost, query storage cost and query processing cost is included by the structure implementation parameters. The structure select the best cost efficient materialize views to optimize the query processing time thereby resulting efficient data warehousing system. This gives the suggestion regarding best view selection for materialization and the effective incremental batch method for materialized view maintenance.

The first approach is for materialized view selection, the essential constraints for materialize view selection are: query frequency, query processing time and storage space[8]. The presented recommended methodology determines which queries are more advantageous using combination of query frequency, processing and storage cost for the creation of materialized view so as to achieve the quick query processing time.

3.4 Materialized view Size Estimation using Sampling

Materializing these summary views can minimize query response time but as the number of views increases exponentially with the number of dimensions, all the views cannot be materialized. However under storage constraint, they must be able to select an optimal number of views to be materialized. To determine the views to be materialized, a Data warehouse administrator may not be able to afford the necessary time to govern the number of rows (size) in each view. Counting actual number of rows represent in each view takes considerable time. They explore the use of sampling to approximate the size of views[9]. It proposes a hybrid estimator that takes into account the degree of skew in the data and combines estimators to estimate the size of the view more accurately. The view size has been used by the proposed hybrid estimator in tpc-h benchmark and the results show better estimation results as compared to individual estimators.

In sequence to apply any problem solving approach, the first step is to know the parameters of the problem occurrence. To completely specify a materialized view selection problem (MVS), one must know the number of rows present in each view and access frequency of each view[9]. Determining access prevalence of each view is a managerial circumspection. The culmination of the contribution of this approach is to apply the concept of sampling based row estimation in a relational table to approximate the size of the views in a data cube taking into account the degree of data skew.

4. COMPARISION BETWEEN TECHNIQUES

Table 1. Comparison between Methods

Techniques	**Description**
View Selection	Materialization of all possible views is not recommended due to memory space and time constraints. It is reduce space and time.
View Adaption	View adaption problem one of the factors that contribute to the changes in a materialized view is rewriting of views that leads to changes in the original view definition itself.
View Synchronization	View synchronization technique changes the view definition when the structure of its base relations changes.
View Maintenance	It is incrementally updates a view by evaluating the changes to be incorporated in the view.

5. CONCLUSION

In this paper we have presented an analysis of different technique being proposed by various researchers to deal with the materialized views in data warehouse namely- view adaptation, view synchronization, view selection and view maintenance. They are all schema object base views techniques. The view selection, view Adaption, view synchronization, view maintenance of to schema object base is one of the most important issues in designing a data warehouse. So as to achieve the best combination of good query response where query processing cost should be decrees in a given storage space constraints. The space constraint is the most important factor while selecting the views to be schema object base. We are taking different parameters for all techniques and analysis. By using these techniques reduce query execution cost, spaces reduce and it is also maintaining large data set.

6. REFERENCES

[1] Amol Deshpande, Lisa Hellerstein "Flow Algorithms for Parallel Query Optimization" IEEE 2008.

[2] Joshi Janki, "An Analysis on Query Optimization in Distributed Database" International Journal of Modern Trends in Engineering and Research 2014.

[3] T.Nalini, Dr. A.Kumaravel, Dr.K.Rangarajan "A comparative study analysis of materialized view for selection cost" International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.1, February 2012.

[4] A. Lee, A. Nica, and E. Rundensteiner, "The EVE approach view synchronization in dynamic distributed environments", In IEEE Transactions and Data Engineering, 14, 2002.

[5] Garima Thakur, Anjana Gosain " A Comprehensive Analysis of Materialized Views in a Data Warehouse Environment" (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 5, 2011

[6] Deepika Kirti, Jaspreeti Singh "Assortment of Materialized View: A Comparative Survey in Data Warehouse Environment"International Journal of Computer Applications (0975 – 8887) Volume 96– No.7, June 2014

[7] Ravindra N. Jogekar, Ashish Mohod "Design and Implementation of Algorithmsfor Materialized View Selection and Maintenance in Data Warehousing Environment" International Journal of Emerging Technology and Advanced Engineering Volume 3, Issue 9, September 2013

[8] S. Chen, X. Zhang, and E. Rundensteiner, "A compensation based approach for view maintenance in distributed environments", In IEEE transactions and data engineering, 18, 2006.

[9] Madhu Bhan, T.V.Suresh Kumar, K.Rajanikanth "Materialized view size estimation using sampling" IEEE International Conference on Computational Intelligence and Computing Research 2013 IEEE