International Journal of Information Sciences and Application.
ISSN 0974-2255 Volume 4, Number 1 (2012), pp. 35-43
© International Research Publication House
http://www.irphouse.com

An Approach for Selection and Maintenance of Materialized View in Data Warehousing

Sanket Patel and Deepak Dembla

Department of Computer, Arya Institute of Engineering and Technology, Jaipur, India
E-mail: it.sanketpatel@gmail.com

Abstract

Quick response time and accuracy are important factors in the success of any database. In large databases particularly in distributed database, query response time plays an important role as timely access to information and it is the basic requirement of successful business application. A data warehouse uses multiple materialized views to efficiently process a given set of queries. The materialization of all views is not possible because of the space constraint and maintenance cost constraint. Materialized views selection is one of the crucial decisions in designing a data warehouse for optimal efficiency. Selecting a suitable set of views that minimizes the total cost associated with the materialized views is the key component in data warehousing. Materialized views are found useful for fast query processing. This paper gives the results of proposed tree based materialized view selection algorithm for query processing. In distributed environment where database is distributed the node on which query should get executed also plays an important role. This paper also proposes node selection algorithm for fast materialized view selection in distributed environment. It found that the proposed algorithm performs well as compare to other materialized view selection strategies.

Keyword: Data Warehousing, Query Processing Cost, Storage Space. View Materialization, View Selection, View-Maintenance

Introduction

A basic requirement for the success of a data warehouse is the ability to provide decision makers with both accurate and timely consolidated information as well as fast query response times. For this purpose, a common method that is used in practice for providing higher information and best response time is the concept of materialized views, where a query is more quickly answered. One of the most important decisions in designing data Warehouse is selecting views to materialize for the purpose of efficiently supporting the decision making. The view selection problem defined is to select a set of derived views to materialize that minimizes the sum of total query response time & Maintenance of the selected views. So the goal is to select an appropriate set of views that minimizes total query response time and also maintains the selected views [1, 13]. The decision "what is the best set of views to materialize?" must be made on the basis of the system workload, which is a sequence of queries and updates that reflects the typical load on the system. One simple criterion would be to select a set of materialized view that minimizes the overall execution time of the workload of queries.

In this paper two algorithms are proposed. First is tree based materialized view selection, in which views are selected at the time of query processing. Second is node selection, which selects nodes in the distributed environment for the execution of the query. In next section various recent past work that has been carried out in the field of materialized view selection and their utilization for the query processing are stated. The proposed algorithm and its implementation details are explained in Section 3. The experiment results that are obtained after the implementation of algorithm are stated and discussed in Section 4. The work that has been carried out is concluded in last section.

Related Work

The distributed model is quickly becoming the preferred medium for file sharing and distributing data over the Internet. A distributed network consists of numerous peer nodes that share data and resources with other peers on an equal basis. Unlike traditional client-server models, no central coordination exists in a distributed system; thus, there is no central point of failure. Distributed networks are scalable, fault tolerant, and dynamic, and nodes can join and depart the network with ease. The most compelling applications on distributed systems to date have been file sharing and retrieval. For example, P2P systems such as Napster [2, 12] and KaZaA [3], are principally known for their file sharing capabilities, for example, the sharing of songs, music, and so on. Furthermore, researchers have been interested in extending sophisticated infrared (IR) techniques such as keyword search and relevance retrieval to distributed databases.

It has been observed that in most typical data analysis and data mining applications, timeliness and interactivity are more important considerations than accuracy; thus, data analysts are often willing to overlook small inaccuracies in the answer, provided that the answer can be obtained fast enough. This observation has been the primary driving force behind the recent development of approximate query processing techniques for aggregation queries in traditional databases and decision support systems [4], [5]. Numerous approximate query processing techniques have been developed: The most popular ones are based on random sampling, where a small random sample of the rows of the database is drawn, the query is executed on this small sample, and the results are extrapolated to the whole database. In addition to simplicity of implementation, random sampling has the compelling advantage that, in addition to an estimate of the aggregate, one can also provide confidence intervals of the error, with high probability. Broadly, two types of sampling-based approaches have been investigated: 1) pre-computed samples, where a random sample is pre-computed by scanning the database and the same sample is reused for several queries and 2) online samples, where the sample is drawn "on the fly" upon encountering a query. So the selection of these random samples in distributed environments for query processing is addressed in [6]. An efficient implementation of materialized sample view is difficult. The primary technical contribution is given in [7] in terms of index structure called the Append ability, Combinability, and Exponentiality (ACE) Tree, which can be used for efficiently implementing a materialized sample view. Such a view, stored as an ACE Tree, has the following characteristics:

It is possible to efficiently sample (without replacement) from any arbitrary range query over the indexed attribute at a rate that is far faster than is possible by using techniques proposed by Olken [8] or by scanning a randomly permuted file. In general, the view can produce samples from a predicate involving any attribute having a natural ordering, and a straightforward extension of the ACE Tree can be used for sampling from multidimensional predicates.

The resulting sample is online, which means that new samples are returned continuously as time progresses and in a manner such that at all times, the set of samples returned is a true random sample of all of the records in the view that match the range query. This is vital for important applications like online aggregation and data mining.

Finally, the sample view is created efficiently, requiring only two external sorts of the records in the view and with only a very small space overhead beyond the storage required for the data records. Note that although the materialized sample view is a logical concept, the actual file organization used for implementing such a view can be referred to as a sample index, since it is a primary index structure for efficiently retrieving random samples.

The basic structure of ACE tree is given in the Figure 1. $I_{i;j}$ refers to the jth internal node at level i. The root node is labeled with a range $I_{1;1}$:R = [0 – 100], signifying that all records in the data set have key values within this range. The key of the root node partitions $I_{1;1}$:R into $I_{2;1}$:R = [0 – 50] and $I_{2;2}$:R = [51 – 100]. Similarly, each internal node divides the range of its descendents with its own key.

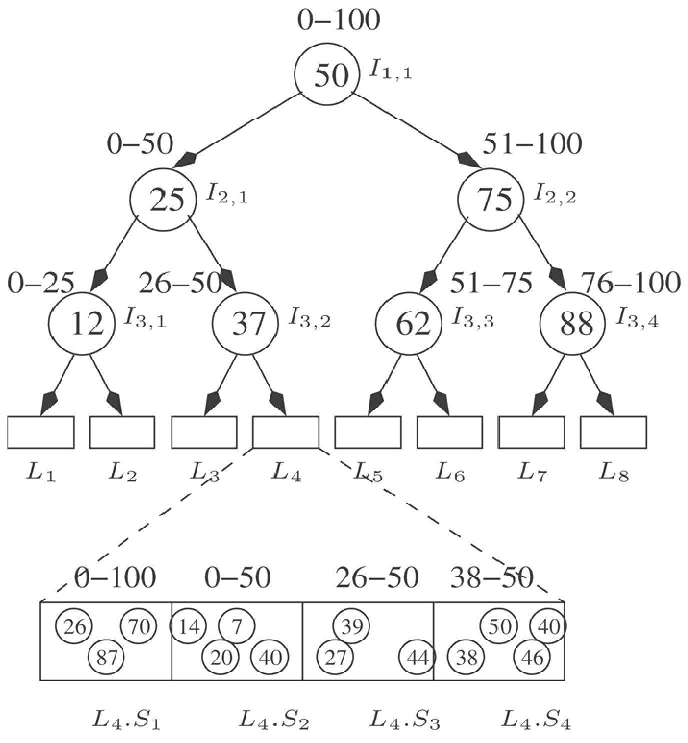


Figure 1: Basic structure of ACE tree

The ranges associated with each section of a leaf node are determined by the ranges associated with each internal node on the path from the root node to the leaf. For example, consider the path from the root node down to leaf node L_4, the ranges that we encounter along the path are 0-100, 0-50, 26-50, and 38-50. Thus, for L_4, L_4:S_1 has a random sample of records in the range 0-100, L_4:S_2 has a random sample in the range 0-50, L_4:S_3 has a random sample in the range 26-50, whereas L_4:S_4 has a random sample in the range 38-50.

A number of parameters, including users query frequencies, base relation update frequencies, query costs, should be considered in order to select an optimal set of views to be materialized. Heuristic Algorithm (HA) [9] will set materialized views such that the total cost for query processing and view maintenance is minimal by comparing the cost of every possible combination of nodes. HA algorithm determines multiple view processing plans regardless of their query cost. HA may include the best processing plan because HA only works with the optimal plans.

In case of 0-1 Programming Algorithm [10] it considers all possible plans for each query to generate a single optimal view processing plan by applying 0-1 integer programming techniques. This works with all the possible join plan trees, therefore it can definitely get the best view processing plan in terms of query access frequency. In A* Heuristic Algorithm [11,14] , an AND-OR view graph and disk space constraints S is given, to deliver a set of views M that has an optimal query response time such that the total maintenance cost of M is less than by satisfying the constraint S. A* algorithm searches for an optimal solution in search graph.

Harinarayan et al. [15] presented a greedy algorithm for the selection of materialized views so that query evaluation costs can be optimized in the special case of “data cubes”. However, the costs for view maintenance and storage were not addressed in this piece of work. Yang et al. [16] proposed a heuristic algorithm which utilizes a Multiple View Processing Plan (MVPP) to obtain an optimal materialized view selection, such that the best combination of good performance and low maintenance cost can be achieved. However, this algorithm did not consider the system storage constraints. Himanshu Gupta and Inderpal Singh Mumick [17] developed a greedy algorithm to incorporate the maintenance cost and storage constraint in the selection of data warehouse materialized views. “AND-OR” view graphs were introduced to represent all the possible ways to generate warehouse views such that the best query path can be utilized to optimize query.

Ziqiang Wang and Dexian Zhang [18] proposed a modified genetic algorithm for the selection of a set of views for materialization. The proposed algorithm is superior to heuristic algorithm and conventional genetic algorithm in finding optimal solutions. Kamel Aouiche et al. [19] proposed a framework for materialized view selection that exploits a data mining technique (clustering), in order to determine clusters of similar queries. They also proposed a view merging algorithm that builds a set of candidate views, as well as a greedy process for selecting a set of views to materialize.

Proposed Algorithm and Implementation Details

In distributed database environment database is present on various nodes. It may happen that same copy of database is present on multiple nodes. So query execution on each and every node will be cumbersome and time consuming. This is more complicated when materialized views are created for the distributed database. The maintenance and selection of materialized views for query execution is challenging task. Two algorithms are proposed for handling the problem of materialized view maintenance and selection.

The first algorithm is for generation and maintenance of materialized view. The tree based approach is used for creating and maintaining materialized views. Initially all records are arranged in ascending order of their key values. Then the middle record is selected as root element of tree. The records are then split till the threshold doesn’t reach so that the leaf of tree should contain the number of records that will be present in materialized view. Then the materialized view will be created for each leaf node indirectly each leaf represent materialized that has to be created and maintain. The materialized view is selected as per the query the records for which the query is intended the materialized view for those records will be selected for the processing. This minimizes the total execution cost. The selective approach can also be used for creating the materialized views that minimizes the storage cost.

The second algorithm is for node selection. This algorithm decides the nodes in the distributed environment for which materialized view should be created, updated or to be maintained. The random walk algorithm is used as base for designing the node selection algorithm and gossip protocol is used to find the best set of the nodes.

Algorithm 1: Tree based materialized view creation and maintenance
r: Threshold for number of records that should be kept in materialized view

Inputs

- R: Total records in database
- m: Number of nodes to visit

Output

- S: Set of Materialized views

```
Begin
    1.  Arrange R in an ascending order of their key values
    2.  Select middle record as a root node
    3.  For all the records in databases available on m
    4.  If number of records in leaf < r
    5.  Split the number of records in equal set
    6.  Else create materialized view for the records which are present in leaf node.
    7.  Add the materialized view in view set
End
```

Algorithm 2: For node selection

M:	Total number of nodes in network
M:	Number of nodes to visit
J:	jump size for randomly selecting nodes
T:	max tuples to be processed per node

Inputs

Q:	Query with selection condition
Sink:	Node where query is initiated
Output:	Query result to Sink (node where query is initiated)

```
Begin
    1.  Check number of active nodes
    2.  If number of nodes = 1
    2.1 Execute query on that node
    3.  Else randomly select the nodes
    4.  Curr = Sink; Hops = 1;
    5.  While (Hops < j * m ) {
    6.  If (Hops % j)
    7.  Visit (Curr);
    8.  Hops ++;
    9.  Curr = random adjacent node
    10. }
    11. Visit (Curr){
    12. If (# tuples of Curr ) <= t){
```

```
    13. Execute Q on all tuples
    14. Else
    15. Execute Q on t randomly sampled tuples
    16. }
    17. Return  result to Sink
    18. Compute Processing time
    19. Return this result to Sink
End
```

Experiment Results and Discussion

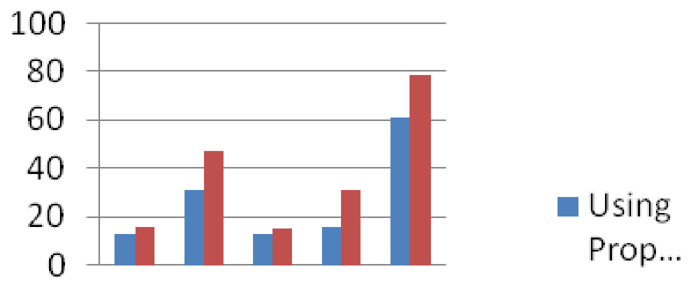
The experiment results are carried out on different databases. BMC, Northwind, Electricity, Web searches and all words databases are used to carry out the experiments of proposed method. The subset of typical user queries is shown in Table 1. The total cost is calculated on the basis of query processing, maintenance and storage cost for the three materialized view strategies the *all-virtual-views* method, the *all-materialized-views* method and the *proposed materialized-views* method.

Table 1 presents the calculation results, from which following observations can be stated: The *all-virtual-views* method requires the highest query processing cost but no view maintenance and storage costs are incurred. The *all-materialized-views* method can provide the best query performance since this method requires the minimum query processing cost. However, its total maintenance and storage expenses are the highest. The *proposed-materialized-views* method requires a lower query processing cost than the *all-materialized-views* method, also its total cost is the least.

Table1: Subset of the query

Strategy	Query Processing Cost	Maintenance Cost	Storage cost	Total Cost
All-virtual-views	16230	0	0	16230
All-materialized-views	1026	2689	1135	4850
Proposed-materialized-views	986	2380	380	3746

The execution time taken by the proposed materialized view algorithm and without using materialized view for various databases is shown in Graph 1. The execution time is given in terms of milliseconds.



Graph 1: Execution time (ms) versus databases

Conclusion

The selection of views to materialize 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 view maintenance cost should be minimized in a given storage space constraints. The proposed algorithms are found efficient as compared to other materialized view selection and maintenance strategies. The total cost, composed of different query patterns and frequencies, were evaluated for three different view materialization strategies: 1) all-virtual-views method, 2) all materialized-views method, and 3) proposed materialized-views method. The total cost evaluated from using the *proposed materialized-views* method was proved to be the smallest among the three strategies. Further, an experiment was conducted to record different execution times of the proposed strategy in the computation of a fixed number of queries and maintenance processes. Again, the *proposed materialized-views* method requires the shortest total processing time.

References

[1] Gorettiv, K.Y., Qing, L. and Ling, F., 2008, "Optimized Design of Materialized Views in a Real-Life Data Warehousing Environment," International Journal of Information Technology, 7(1).

[2] NapsterHomepage, http://www.napster.com,

[3] Kazaa Homepage, 2006, http://www.kazaa.com.

[4] Babcock, B., Chaudhuri, S., and Das, G., 2003, "Dynamic Sample Selection for Approximate Query Processing," proc. 22nd ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03), pp. 539-550.

[5] Gkantsidis, C., Mihail, M., and Saberi, A., 2004, "Random Walks in Peerto-Peer Networks," Proc. IEEE INFOCOM '04.

[6] Benjamin, A., Gautam, D,, Dimitrios, G., and Vana, K., 2007, "Efficient Approximate Query Processing in Peer-to-Peer Networks," IEEE Trans on Knowlwgde and Data Engg., 19(7).

[7] Shantanu, J., and Christopher, J., 2008, "Materialized Sample Views for Database Approximation," IEEE Trans on Knowledge and Data Engg., 20(3).
[8] Olken, F., 1993, "Random Sampling from Databases," Ph.D dissertation.
[9] Choi, C.H., Yu, J.X., and Gou, G., 2002, "What difference heuristic make: maintenance cost view selection revisited," proc. 3rd Int. Conf. on Advances in Web-Age Information Management, Springer-Verlag., pp.313-350.
[10] Yang, J., Karlapalem, K., and Li, Q., 1997, "Algorithms for materialized view design in data warehousing environment," proc. 23rd Int. Conf. on Very Large Data Bases, pp.136-145.
[11] Gang, G., Jeffery, X.Y., and Hongjun, L., 2006, "A* Search: An Efficient and Flexible Approach to Materialized View Selection," IEEE Trans. on Systems, Man and Cybernetics – Part C: Appl. And Reviews, 36(3).
[12] Bazlur, R.A.N.M., and Islam, M.S., 2009, "An Incremental View Materialization Approach in ORDBMS," IEEE Intl. Conf. on Recent Trends in Information, Telecommunication and Computing
[13] Zhou, L., Ge, X., Wang, L., and Shi, Q., 2009, "Research on Materialized View Selection Algorithm in Data Warehouse," IEEE Intl. Conf. on Computer Science-Technology and Applications.
[14] Qingzhou, Z., Xian, S., and Ziqiang, W., 2009, "An Efficient MA-Based Materialized Views Selection Algorithm," IEEE Intl. Conf on Control, Automation and Systems Engineering.
[15] Harinarayan, V., Rajaraman, A., and Ullman, J., 1996, "Implementing data cubes efficiently," Proc. of ACM SIGMOD 1996 International Conference on Management of Data, Montreal, Canada, pp. 205—216.
[16] Yang, J., Karlapalem, K., and Li, Q., 1997, "A framework for designing materialized views in data warehousing environment," Proc. of 17th IEEE International conference on Distributed Computing Systems, Maryland, U.S.A.
[17] Gupta, H., 1997, "Selection of Views to Materialize in a Data Warehouse," Proc. of International Conference on Database Theory, Athens, Greece.
[18] Ziqiang, W., and Dexian, Z., 2005, "Optimal Genetic View Selection Algorithm Under Space Constraint," International Journal of Information Technology, vol. 11, no. 5, pp. 44 – 51.
[19] Aouiche, K., Jouve, P., and Darmont, J., 2006, "Clustering-based materialized view selection in data warehouses." In ADBIS'06, volume 4152 of LNCS, pages 81–95.