Optimized Cost Effective Approach for Selection of Materialized Views in Data Warehousing

B.Ashadevi
Assistant Professor, Department of MCA
Velalar College of Engineering and Technology
Erode, Tamil Nadu, India
aashadeviphd@yahoo.com
and
Dr.R.Balasubramanian
Dean of Basic Sciences
Velammal Engineering College
Chennai, Tamil Nadu, India
Balasubramanian_R07@dataone.in

ABSTRACT

A data warehouse efficiently processes a given set of queries by utilizing the multiple materialized views. Owing to the constraint on space and maintenance cost, the materialization of all views is unfeasible. One of the critical decisions involved in the process of designing a data warehouse for optimal efficiency, is the materialized views selection. The primary goal of data warehousing is to select a suitable set of views that minimizes the total cost associated with the materialized views. In this paper, we have presented a framework, an optimized version of our previous work, for the selection of views to materialize, for a given storage space constraints, which intends to achieve the best combination of good query response, low query processing cost and low view maintenance cost. All the cost metrics associated with the materialized views selection that comprise the query execution frequencies, base-relation update frequencies, query access costs, view maintenance costs and the system's storage space constraints are considered by this framework. This framework optimizes the maintenance, storage and query processing cost as it selects the most cost effective views to materialize. Thus, an efficient data warehousing system is the outcome.

Keywords: Data Warehousing, Views, Materialization, View Selection, View-Maintenance, Query processing cost, Storage space.

1. INTRODUCTION

The accumulation of data has led to the recent availability of outsized archives of data in industry and organization. The decision making process is faced by critical problems due to the employment of these bulk data. These problems can be managed by the developing new data models and decision support systems. Warehousing is an emerging technique that retrieves the data from distributed autonomous probably heterogeneous information sources and integrates the retrieved data [1]. On-Line Analytical Processing and Decision Support Systems utilize the large volume of extracted and summarized data stored in an information base referred as a data warehouse [2]. The data warehousing technologies is the basis for the effective embarking of many industries, for instance, manufacturing financial services, transportation, telecommunications, utilities and healthcare.

In order to collect data from many data sources, a data warehouse uses an update-driven approach that communicates through networks both locally and internationally. A solid platform of consolidated historical data is provided for analysis by the data warehouse system and it also distributes such analysis to local and remote users [3]. In order to provide effective solution for the queries posted to the data warehouse, the intermediate results obtained in the query processing are stored in the data warehouse. This can avert the access of the original data sources by the users [4]. A view is a derived relation defined in terms of base (stored) relations. A data warehouse holds multiple views and we have referred the materialized views as the views stored in the data warehouse.

Materialized views are physical structures that pre-compute the intermediary results, thereby improving data access time. However, additional storage space and maintenance overhead when refreshing the data warehouse is necessitated by the employment of materialized view [5]. Owing to the direct availability of integrated information at the warehouse with differences already resolved, data warehouse has the ability to answer queries and perform analysis efficiently and quickly [1]. The data warehouse research community provides effective solutions for the problem of representing data in a form suitable for analytical queries, however other performance issues such as, query response time for a given aggregated query, view maintenance time, etc are not entirely dealt with.

The materialization of views is the most important ordeal in data warehousing. It is impossible to materialize all possible views as large computation and space is necessitated. Consequently, the primary concern in data warehousing is the "view selection problem" that deals with the selection of suitable set of views to materialize that strikes a stability among computational cost and increased query performance

[6]. In a dynamic environment, the selection of appropriate set of views to materialize necessitate consideration of additional factors, hence, it is a demanding task. The selection of the materialized views is affected by numerous factors. Thus, the process of selecting the suitable views to materialize in warehouse implementation is a critical issue.

The problem of materialized view selection is NP-hard [6]. Numerous methods in literature like the ones in [5], [6 -13] have attempted to solve the materialized view selection problem. Lately, evolutionary approaches (genetic algorithms) have been projected for achieving improved results in the view maintenance and query processing costs [3], [4], [14], [15]. Nevertheless, some problems arise due to the probability of impractical solutions. The constraint of storage space was addressed in [10]. In [6], [16] and [3], the constraint of maintenance cost was considered. A genetic algorithm without any restriction is presented in [4]. A genetic algorithm to tackle this problem under the constraint maintenance cost with the introduction of a function of penalty is projected in [13]. A genetic algorithm that does not utilize function penalty is proposed in [3]. In [17], the genetic algorithm is enhanced further by altering the genetic operators and the restoring scheme of impractical solutions. A majority of these approaches utilize AND – DAG for the query representation .The AND-DAG utilizes tree based structure resulting in computational complication and increased traversal time.

The research is focused on developing a framework for choosing views to materialize in order to achieve improved query response in low time by the reduction of the total cost involved with the materialized views. All the cost metrics associated with materialized views like the query execution frequency, query access cost, base-relation update frequency, view maintenance cost and the system's storage space constraints are utilized by the proposed framework. Existing materialized views are maintained by the system from time to time by confiscating views with low access frequency and high storage space. The queries having high access frequencies are chosen for the view selection problem. The intermediary views in the queries are represented in a simple format, an optimized form of our previous work [19], rather using AND-DAG which uses tree based structure resulting in computational complexity and more traversal time. The proposed format results in reduced computational complexity. An algorithm is projected for choosing the views to materialize on basis of their weightage in the provided query set.

The rest of the paper is organized as follows; Section 2 presents a brief review of related works in materialized views selection. Section 3 presents the proposed framework for materialized views selection. The experimental results are given in Section 4 and conclusions are summed up in Section 5.

2. RELATED WORKS

The problem of finding views to materialize to answer queries has traditionally been studied under the name of view selection. Its original motivation comes up in the context of data warehousing.

Harinarayan et al. [7] 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. [8] 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 [9] 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 response time. Amit Shukla et al. [10] proposed a simple and fast heuristic algorithm, PBS, to select aggregates for precomputation. PBS runs several orders of magnitude faster than BPUS, and is fast enough to make the exploration of the time-space tradeoff feasible during system configuration. Himanshu Gupta and Inderpal Singh Mumick [6] developed algorithms to select a set of views to materialize in a data warehouse in order to minimize the total query response time under the constraint of a given total view maintenance time. They have designed approximation algorithms for the special case of OR view graphs.

Chuan Zhang and Jian Yang [15] proposed a completely different approach, Genetic Algorithm, to choose materialized views and demonstrate that it is practical and effective compared with heuristic approaches. Sanjay Agrawal et al. [11] proposed an end-to-end solution to the problem of selecting materialized views and indexes. Their solution was implemented as part of a tuning wizard that ships with Microsoft SQL Server 2000.

Chuan Zhang et al. [4] explored the use of an evolutionary algorithm for materialized view selection based on multiple global processing plans for queries. They have applied a hybrid evolutionary algorithm to solve problems. Minsoo Lee and Joachim Hammer [14] proposed an efficient solution to the maintenance-cost view selection problem using a genetic algorithm for computing a near-optimal set of views used to search for a near optimal solution.

Panos Kalnis et al. [12] proposed the application of randomized search heuristics, namely Iterative Improvement and Simulated Annealing, which select fast a sub-optimal set of views. The proposed method provided near-optimal solutions in limited time, being robust to data and query skew. Jeffrey Xu Yu et al. [3] proposed a new constrained evolutionary algorithm for the maintenance-cost view-selection problem. Constraints were incorporated into the algorithm through a stochastic ranking procedure. No penalty functions were used. Ziqiang Wang and Dexian Zhang [17] 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. [5] 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.

3. FRAMEWORK FOR SELECTION OF MATERIALIZED VIEWS

This section explains the proposed cost effective framework for materialized view selection. The proposed framework exploits all the cost metrics associated with materialized views such as query frequency, query access cost, base-relation update frequency, view maintenance cost and the system's storage space constraints. The materialized view selection problem can be described as follows: Given a set of queries Q and a quantity S (available storage space) and maintenance time MT and existing materialized views Mv, the view selection problem is to select a set of views M to be materialized, that minimize total cost associated with materialized views under storage space and maintenance cost constraints. The storage space constraint is the space which should not be exceeded by materializing the views. The maintenance cost constraint is the total time which should not be exceeded while maintaining the materialized views.
The framework sustains existing materialized views periodically by removing views with low access frequency and high storage space. The queries with high access frequencies are selected for the view selection problem. The intermediary views are represented in a comparatively simpler format than that of our previous work, which in itself was an enhancement over the conventional representation with the aid of AND-DAG that makes use of a tree based structure resulting in computational complexity and additional traversal time. An algorithm is proposed for the selection of views to materialize based on their weightage in the given query set and storage space. Then the query access cost and maintenance cost of selected views are calculated. The total cost of each view is calculated and views with optimum cost under the maintenance and space constraints are selected for materialization. The proposed framework is discussed detailed in the following sub-sections.

Preservation of Existing Materialized Views

This sub-section details the preservation of the existing materialized views. Before selecting new views for materialization, the existing materialized views are sustained based on their access frequency and storage space. The steps for the above process are given in Algorithm1.

Assumptions:

M_V	→ Vector of Materialized views
N	→ Total no of materialized views
MS	→ Memory size of materialized views
$Thres$	→ Threshold value
AF	→ Access frequency of Materialized views

Algorithm 1:

```
for each Materialized View in $M_V$
        $W = 2\log(AF) - \log(MS)$
        if (W < Thres) then
                Remove current Materialized view;
        end if
end for
```

The above algorithm removes the materialized views with low access frequency and high storage space for the materialization of new views.

Weightage Based View Selection

This sub-section details the initial selection of views based on their weightage in the given query set and storage space. Instead of selecting all the queries, the queries which have high access frequency are selected for the view selection problem. The queries are selected from the given query set using Algorithm 2.

Assumptions:

Q	→ Given Set of Queries
Q_{AF}	→ Access Frequency of Queries
Φ	→ Threshold value
SQ	→ Vector of selected queries

Algorithm 2:

```
for each query in Q
        if $(Q_{AF} > \Phi)$ then
                Add query to vector SQ ;
        end if
end for
```

The queries having access frequency greater than the threshold value Φ are selected for materialized view selection problem. After that the conditional clauses in each query are represented in a simple using Algorithm 3.

Assumptions:

SQ	→ Selected set of Queries
Q_C	→ 2D Array of conditional clauses
Q_{SV}	→ 2D Array of integer values of Q_C

Algorithm 3:

```
for each query in SQ
        if the query has conditional clauses then
        $Q_C[i] = conditional\ Clause$ (CQ)
        end if
end for
```

Each distinct conditional clause in Q_C is mapped to an integer value and the count of each distinct clause is calculated using Algorithm 4.

Assumptions:

DCC	→ Distinct conditional clauses
CC	→ Count of conditional clause

Algorithm 4:
The conditional clauses in each query are represented in 2-D format using Algorithm 3. This 2-D representation is converted into 1-D representation and their counts are taken simultaneously for further processing. The algorithm for the above is as follows:

```
Set index = 0;
for each(i) row in $Q_c$
        for each(j) conditional clause $C_c$ in row
            $if((DCC \cap C_c) == \Phi)$
                $DCC << C_c;$
                $CC << 1;$
            else
                $index = DCC[C_c];$
                $CC[index] = CC[index] + 1;$
            end if
        end for
end for
```

Then the views are selected based on their weightage in the given query set and storage space using Algorithm 5. Then views with weightage greater than a threshold value α are selected for further process.

Assumptions:

M_U	→ Vector of Storage space needed to store result of conditional clause
M_{Tot}	→ Total storage space needed
CC_{Tot}	→ Total Count
SV	→ Selected set of views

Algorithm 5:

```
for each conditional clause in DCC
        $F1 = CC / CC_{Tot};$
        $F2 = (1 - (M_U / M_{Tot}));$
        $W = 2\log(F1) + \log(F2);$
        $If(W > \alpha);$
            Add current conditional clause based
            view to $SV$ for further process
        end if
end for
```

Query Processing Cost
The cost of query processing is query frequency multiplied by the cost of query access from the materialized views. The query processing cost of each view from SV is calculated using the following formula.

$$QP_{COST} = 1/\sum_{i=1}^{N} Freq * Ca(V)$$

Where N is the total no of queries, $Freq$ is the frequency of query and $Ca(V)$ is the cost of access for query q using view V .

View Maintenance Cost
View maintenance is the process of updating pre-computed views when the base fact table is updated. The maintenance cost for materialized view is the cost used for refreshing this view whenever a change is made to the base table. The maintenance cost is calculated using update frequency and the priority value of the base table. A priority value in the range 1 – 10 is assigned for each base table based on its importance. The maintenance cost is calculated using Algorithm 6.

Assumptions:

P	→ Priority of Base tables
UF	→ Update frequency of Base tables

Algorithm 6:

```
for each view in SV
        for each base table
        $VM_{COST}[i] = 1/(P[i] * (1/UF[i]));$
        end for
end for
```

Materialized View Selection
The total cost of each view is calculated by summing the query processing cost and maintenance cost. Then the views are sorted in ascending order based on their total cost.

$$TotCost = QP_{COST} + VM_{COST}$$

Then the views with minimal cost whose maintenance time and storage space falls within the given constraints are selected for materialization.

4. EXPERIMENTAL RESULTS

In this section, we have presented the results of our experimental analysis. We have implemented all the algorithms of our proposed approach in Java. The Algorithm 1 has successfully removed the existing materialized views with low access frequency and high storage space and thus freed the space for the materialization of new views. The Algorithm 2 has successfully selected the queries with high access frequencies for the view-selection problem. The conditional clauses from each selected query were extracted by Algorithm 3. We have optimized our previous algorithm [19] into algorithm 4 in the proposed work. From the available views, some views were initially selected based on Algorithm 5. We can conclude that, our framework finally selects view with minimum cost for materialization under the storage space constraints and maintenance cost constraints by considering all the cost metrics associated with the materialized views.

We have compared optimized CEMS (Cost Effective approach for Materialized view Selection) against CEMS [19] with the aid of time. The optimized CEMS consumes less time than the CEMS algorithm. The experiments are carried out with files of varying query sizes and the time taken to materialize the queries is estimated. The results of our experiments have been clearly shown in the table and the analysis is presented in the graph.

In Table 1, the size of the files having queries and the time taken to materialize those queries in both algorithms is given. The graphical representation (Figure 1, 2) shows that the optimized algorithm is better than our previous work in terms of time.

Table 1: comparative results of execution time for CEMS and optimized CEMS

File size (KB)	CEMS (Sec)	Optimized CEMS (Sec)
0.5	0.266	0.188
1	0.297	0.25
1.5	0.39	0.358
2	5.125	5.016
2.5	38.204	38.047
3	51.828	51.688
3.5	105.207	104.954
4	130.984	130.641
4.5	171.14	170.766

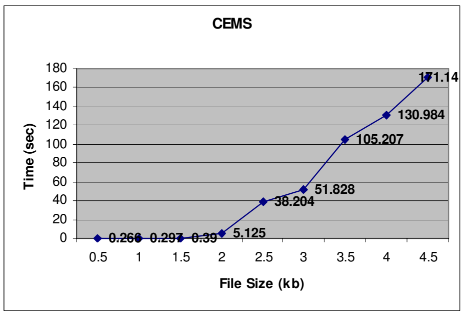

Figure 1: Time Analysis Graph of CEMS [19]

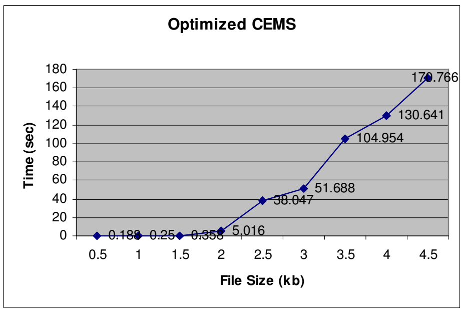

Figure 2: Time Analysis Graph of Optimized CEMS

5. CONCLUSION

The selection of views to materialize is one of the most important issues in designing a data warehouse. The view-selection problem has been addressed in this paper by means of taking into account the essential constraints: maintenance cost and storage space. In this paper, we have presented a framework, which is an optimized version of our previous work, for selecting views to materialize so as to achieve the best combination of good query response, low query processing cost and low view maintenance cost in a given storage space constraints. The presented framework considered all the cost metrics associated with materialized views such as query execution frequencies, base-relation update frequencies, query access costs, view maintenance costs and the system's storage space constraints. The most cost effective views have been selected for materialization by the framework and the maintenance, storage and query processing cost of the views have been optimized. We have compared the results with our previous work in terms of time.

6. REFERENCES

[1] Y. Zhuge, H. Garcia-Molina, J. Hammer, and J. Widom, "View Maintenance in a Warehousing Environment." In Proceedings of the ACM SIGMOD Conference, San Jose, California, May 1995.

[2] S. Chaudhuri and U. Dayal. "An Overview of Data Warehousing and OLAP Technology". SIGMOD Record, vol: 26, no: 1, pp: 65-74, 1997.

[3] J. X. Yu, X. Yao, C. Choi and G. Gou. Materialized view selection as constrained evolutionary optimization. IEEE Transactions on Systems, Man and Cybernetics, Part C, vol: 33, no: 4, pp: 458–467, 2003.

[4] C. Zhang, X. Yao, and J. Yang. An evolutionary Approach to Materialized View Selection in a Data Warehouse Environment. IEEE Transactions on Systems, Man and Cybernetics, vol. 31, no.3, pp. 282–293, 2001.

[5] K. Aouiche, P. Jouve, and J. Darmont. Clustering-based materialized view selection in data warehouses. In ADBIS'06, volume 4152 of LNCS, pages 81–95, 2006.

[6] H. Gupta, I.S. Mumick, Selection of views to materialize under a maintenance cost constraint. In Proc. 7th International Conference on Database Theory (ICDT'99), Jerusalem, Israel, pp. 453–470, 1999.

[7] V. Harinarayan, A. Rajaraman, and J. Ullman. “Implementing data cubes efficiently”. Proceedings of ACM SIGMOD 1996 International Conference on Management of Data, Montreal, Canada, pages 205--216, 1996.

[8] J.Yang, K. Karlapalem, and Q. Li. “A framework for designing materialized views in data warehousing environment”. Proceedings of 17th IEEE International conference on Distributed Computing Systems, Maryland, U.S.A., May 1997.

[9] H. Gupta. “Selection of Views to Materialize in a Data Warehouse”. Proceedings of International Conference on Database Theory, Athens, Greece 1997.

[10] A. Shukla, P. Deshpande, and J. F. Naughton, “Materialized view selection for multidimensional datasets,” in Proc. 24th Intl. Conf. Very Large Data Bases, pp. 488–499, 1998.

[11] S. Agrawal, S. Chaudhuri, and V. Narasayya, “Automated Selection of Materialized Views and Indexes in SQL Databases,” Proceedings of International Conference on Very Large Database Systems, 2000.

[12] P. Kalnis, N. Mamoulis, and D. Papadias, “View Selection Using Randomized Search,” Data and Knowledge Eng., vol. 42, no. 1, 2002.

[13] Gupta, H. & Mumick, I., Selection of Views to Materialize in a Data Warehouse. IEEE Transactions on Knowledge and Data Engineering, vol: 17, no: 1, pp: 24-43, 2005.

[14] M. Lee and J. Hammer, Speeding up materialized view selection in data warehouses using a randomized algorithm, International Journal of Cooperative Information Systems, 10(3):327–353, 2001.

[15] C. Zhang and J. Yang, “Genetic algorithm for materialized view selection in data warehouse environments,” Proceedings of the International Conference on Data Warehousing and Knowledge Discovery , LNCS, vol. 1676, pp. 116–125, 1999.

[16] C. -H. Choi, J. X. Yu, and G. Gou, “What difference heuristics make: Maintenance-cost view-selection revisited,” in Proc. Third Int. Conf. Web-Age Information Management, 2002.

[17] Ziqiang Wang and Dexian Zhang, Optimal Genetic View Selection Algorithm Under Space Constraint, International Journal of Information Technology, vol. 11, no. 5, pp. 44 - 51, 2005.

[18] Gang Gou; Yu, J.X.; Hongjun Lu., "A* search: an efficient and flexible approach to materialized view selection Systems," IEEE Transactions on Man, and Cybernetics, Part C: Applications and Reviews, Vol. 36, no. 3, May 2006 pp: 411 - 425.

[19] B.Ashadevi, R.Balasubramanian, "Cost Effective Approach for Materialized Views Selection in Data Warehousing Environment", proc. of the International Journal of Computer Science and Network security Vol. 8, No. 10, pp. 236-242, 2008.