SELECTION OF MATERIALIZED VIEW USING QUERY OPTIMIZATION IN DATABASE MANAGEMENT : AN EFFICIENT METHODOLOGY

Mr. P. P. Karde[1] and Dr. V. M. Thakare[2]

[1]Department of IT, HVPM's College of Engg. & Tech, Amravati, (M.S),India
P_karde@rediffmail.com
[2]Post Graduate Deptt. of Computer Science, SGB, Amravati University, Amravati,
vilthakare@yahoo.co.in

ABSTRACT

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. Quick response time and accuracy are important factors in the success of any database. The materialization of all views is not possible because of the space constraint and maintenance cost constraint. Selection of Materialized views is one of the most important 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 and is the key component in data warehousing. Materialized views are found to be very 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 over the nodes on which query should get executed and also plays an important role. This paper also proposes node selection algorithm for fast materialized view selection in distributed environment. And finally it is found that the proposed methodology performs better for query processing as compared to other materialized view selection strategies.

KEYWORDS:

Data warehousing, Materialize views, Net benefit, Query cost, Storage cost, View maintenance, View selection

1. 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, 25]. 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.

A view is defined as a function from a set of base tables to a derived table and the function is recomputed every time the view is referenced. On the other hand, a materialized view is like a cache *i.e.*, a copy of data that can be accessed quickly. Utilizing materialized views that incorporate not just traditional simple SELECT-PROJECT-JOIN operators but also complex online analytical processing operators play crucial role to improve the OLAP query performance. Materialized views are useful in applications such as data warehousing, replication servers, data recording systems, data visualization and mobile systems [2, 3, 4]. In certain situation, it is more profitable to materialize a view than to compute the base tables every time the view is queried. Materializing a view causes it to be refreshed every time a change is made to the base tables that it references. It can be costly to rematerialize the view each time a change is made to the base tables that might affect it. So it is desirable to propagate the changes incrementally *i.e.*, the materialized view should be refreshed for incremental changes to the base tables. In the last few years, several view maintenance methods have been designed and developed to obtain an efficient incremental view maintenance plan [5]. In this paper a methodology has been presented. First is tree based materialized view selection, in which views are selected at the time of query processing. Second is node selection, in which the nodes are selected in the distributed environment for the execution of faster query performance. 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 4 The experimental results that are obtained after the implementation of algorithm are stated and discussed in Section 5. The work that has been carried out is concluded in last section.

2. MATERIALIZED VIEW MANAGEMENT AND SELECTION

Materialized View Management & Selection Approach

The motivation for using materialized views is to improve performance but the overhead associated with materialized view management can become a significant system management problem. The common materialized view management activities include: identifying which materialized view to create; indexing the materialized view; ensuring that all materialized views and materialized view indexes are refreshed properly each time the database is updated; checking which materialized views have been used; determining how effective each materialized view has been on workload performance; measuring the space being used by materialized views; determining which existing materialized views should be dropped; archiving old detail and materialized view data that is no longer useful [6,28].

The view selection problem is to choose a set of views to materialize in order to achieve the best query performance for a given query workload. Typically view selection is under a space constraint, and / or a maintenance cost constraint [7, 8, 28]. Unlike answering queries using views that need to handle adhoc queries, in view selection scenarios, the queries are known. Hence, most view selection algorithms start from identifying common sub-expressions among queries. These common sub expressions serve as the candidates of the materialized views. One fundamental practical issue with view selection is that there are many possibly competing factors to be considered during the view selection phase, such as view selectivity, query complexity, database size, query performance, update performance etc.

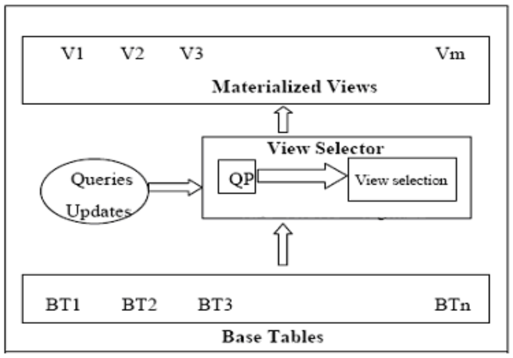


Figure 1: View Materialization Process

In the above architecture the view selector interacts with the query processor (QP). Based on the query processing plan it applies the notion of view relevance to select the views for a given set of queries.

3. RELATED WORK

Harinarayan et al. [10] 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. [11] 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 [12] 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.

In case of 0-1 Programming Algorithm [13] 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 [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.

Ziqiang Wang and Dexian Zhang [15] 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. [16] 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.

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 [17] and KaZaA [18] 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 [19, 20]. 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 [21].

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) [22] 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.

An efficient implementation of materialized sample view is difficult. The primary technical contribution is given in [23] in terms of index structure called the Appendability, Combinability, and Exponentially (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:

1. 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 [24] 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.
2. 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.

3. 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 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.

4. Proposed Algorithms 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. Therefore query execution on each and every node will be cumbersome and time consuming in distributed environment. This becomes 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 proposed algorithms are presented 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 available in materialized view. Then the materialized view is created for each leaf node, indirectly each leaf represents materialized view 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 and only those records will be selected for the processing. This minimizes the total execution time for query processing. 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.

In the following algorithm initially records are arranged in an ascending order of their key values using arrange(R). Then the middle record is selected as a root node. For each record on the available nodes, if the threshold is less than the number of records in leaf node then again split the records in equal sets; otherwise create the materialized view for next available records in the leaf node & add this materialized view in the view set.

Algorithm 1: Tree Based Materialized View Creation and Maintenance

```
r: Threshold for number of records that should be kept in materialized view
P: Root Node
S: Number of records in leaf
```

Inputs:

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

Output:

```
S: Set of Materialized views
```

Begin

```
1.    arrange (R)
2.    N = middle (R)
3.    Repeat
          3.1 If (S > r)
                  Split (S)
          3.2 Else create Materialized_View (S).
4.    Add (View)
5.    Until  R ! =  NULL
```

End

For node selection algorithm, initially it checks the available active nodes from available ‘S’ nodes. If there is only one node then the query will be executed on the same node; otherwise the random nodes will be identified from the available ‘P’ active nodes on which query will get executed.

Algorithm 2: For Node Selection

```
M: Total number of nodes in network
N: Number of Active Nodes
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.    N = Active Nodes (M)
2.    If N = = 1
```

```
          2.1. Execute query on N
3.        Else  m = random (N)
4.        Curr = Sink;  Hops = 1;
5.        While (Hops < j * m ) {
          5.1. If (Hops % j)
               5.1.1.  Visit (Curr);
               5.1.2.  Hops ++;
               5.1.3.  Curr = random adjacent node }

6.        Visit (Curr)
7.        If (# tuples of Curr ) <= t)
          7.1. Execute Q on all tuples
8.        Else
          8.1. Execute Q on t randomly sampled tuples
9.        Return  Result to Sink
10.       Compute Processing Time
11.       Return this Result to Sink
End
```

Cost Analysis

The total cost for materializing views can computed using the following strategy. The proposed algorithm considers query processing cost (for selection, aggregation and joining), view maintenance cost, storage cost, net benefit and storage effectiveness for computing the total cost. The cost is calculated in terms of block size B. The query processing cost in terms of block access is equal to size of materialized view V_i. [25, 28, 1,12]

$$C_B (V_i) = S(V_i)$$

The query cost involving the joining of n dimensional tables with view V_i is given by

$$C_j(V_{d1}, V_{d2},\dots, V_{dn} , V_i) = (S(V_{d1}) + S(V_{d1}) *S(V_i)) + (S(V_{d2}) + S(V_{d2}) *S(V_i)) + \dots + (S(V_{dn}) + S(V_{dn})* S(V_i))$$

To process user's query q_i, which requires not only selection and aggregation of the view, but also the joining of view with other dimension tables, the query cost $C_q(q_i)$ is given by

$$C_q(V_i) = C_B (V_i) + C_j(V_{d1}, V_{d2},\dots, V_{dn} , V_i) = S(V_i) + (S(V_{d1}) + S(V_{d1}) *S(V_i)) + (S(V_{d2}) + S(V_{d2}) *S(V_i)) + \dots + (S(V_{dn}) + S(V_{dn}) *S(V_i))$$

Thus the total Query cost Total (C_{qr}) for processing r user queries is given by

$$Total\,(C_{qr}) = \sum_{i=1}^{r} \left(\mathbf{f}_{qi} * C_q(q_i)\right)$$

The re-computation of each view requires selection and aggregation from its ancestor view V_{ai}, and their joining with n dimension tables. Therefore the maintenance cost is given by

$$C_m(V_i) = C_B (V_{ai}) + C_j(V_{d1}, V_{d2},\dots, V_{dn} , V_{ai}) = S(V_i) + (S(V_{d1}) + S(V_{d1}) *S(V_{ai})) + (S(V_{d2}) + S(V_{d2}) *S(V_{ai})) +$$

$$\ldots + (S(V_{dn}) + S(V_{dn}) * S(V_{ai}))$$

If there are j views which are materialized, the total maintenance cost Total (C_m) for these materialized views is given by

$$Total\ (C_m) = \sum_{i=1}^{j} \left(f_{ui} * C_m(V_i) \right)$$

The cost for storing materialized views depends on the availability of hard disk space. The storage factor U represents the estimated ratio of the storage capacity required by the data warehouse to the availability of hard disk space it is given by

$$U = (\text{Total } (C_{store}) + (1+Q) * Y * S_a) \ / \ \text{Total available storage capacity}$$

Where '$(1+Q) * Y * S_a$' estimates the total increase in storage capacity for accommodation of new data during processing or creation of materialized views. Here Q is the estimated increase rate in data volume per year within data warehouse, Y is the estimated processing cycle of the data warehouse, and S_a is the storage space required to store added new data and their materialized data.

The storage cost of view in terms of data block B is given by

$$C_{store}\ (V_i) = U * S\ (V_i)$$

In most of the today's systems storage space doesn't matter because large amount of hard disk space is available with less prize so in proposed algorithm implementation the value of U=1. Therefore the total storage cost is calculated as

$$C_{store}\ (V_i) = S(V_i)$$

The net benefit and the storage effectiveness can be calculated to determine an optimal set of materialized views. The net benefit of materializing view calculated as follows [26, 27,1]

Net Benefit = Benefit – Maintenance cost –Storage cost

$$(B_i) = \sum_{i=1}^{m} \left(f_{qi}(V_{ni}) \right) * [C_t(V_{ni} <- V_{at}) - C_t\ (V_{ni} <- V_i)])$$

Here, V_{ni} represents one of the descendent views of V_i and m is the total number of descendent views. C_t represents the cost of accessing materialized view. Therefore, the net benefit for materialized view can be calculated as Net (B_i) = B_i - C_m (V_i) - C_{store} (V_i)

The storage effectiveness of views is given by n_i = Net (B_i) / S (V_i).

Consider Total(C_{all}) is the total cost for processing user's queries when no views are materialized in the data warehouse. When the materialized views are used then total cost is given by

$$C_{total} = \text{Total } (C_{all}) - \sum_i \text{Net}\ (B_i)$$

5. EXPERIMENTAL 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 using 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 three materialized view strategies the *all-virtual-views* method, *all-materialized-views* method and the *proposed materialized-views* method.

Table 2 represents the calculation results, from which following observations can be stated: The *all-virtual-views* method requires the highest cost of query processing with no view maintenance and storage costs are incurred. The *all-materialized-views* method can provide the best query performance but highest cost of view maintenance 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 all-*materialized-views* method, also its total cost is also minimized.

Table 1: Subset of user queries

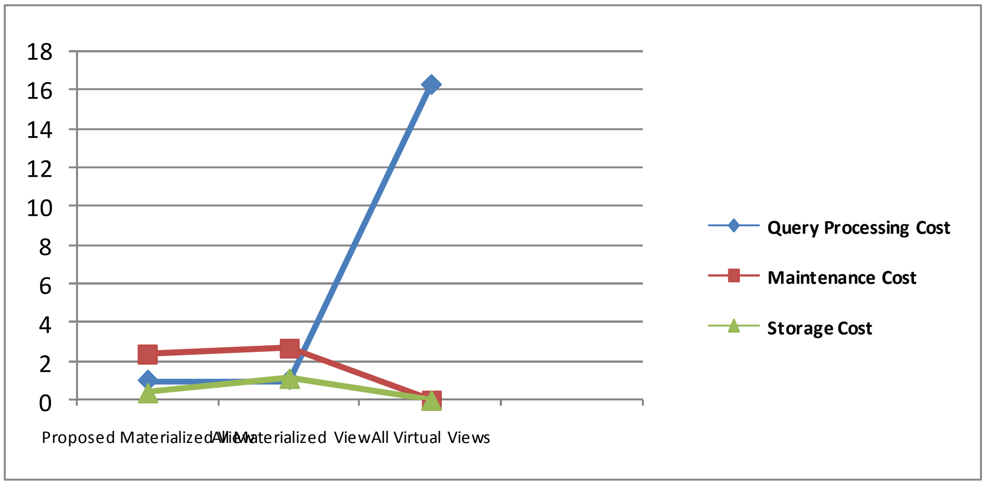
Queries	**Query freq.**	**Views**	**Number of Records in Summary view Table**	**Size (in Bytes)**
SELECT SR, DO, AREA, CUSTOMER, EMTBRANCH, PRINCIPAL, MODEL, CNCCONTROL, MACHINESR, DELYON, STARTON, COMMON, COMMANBY, WARRENTYUPTO, REMARKS, TARGETDT FROM BMC ORDER BY DO;	2	BMC View	4387	289.00
SELECT DIVISIONSTATE, RESIDENTIAL, COMMERCIAL, INDUSTRIAL, TRANSPORTATION, ALLECTORS FROM ELEPRICEPERUSER ORDER BY ALLSECTORS;	1	ELEPRICEP ERUSER View	4660	310.00
SELECT URL, DATE FROM SEARCHES ORDER BY DATE;	1	SEARCHES View	3000	156.00
SELECT PRODUCTID, NAME, DEALER, PURCHASEDATE, QUANTITY, MANUFACTURINGDATE, SOLD, PRODUCTGRPID FROM PRODUCTDETAILS GROUP BY PRODUCTID;	1	PRODUCTD ETAILS View	5564	380.00

Table 2: The Query Processing, Maintenance and Storage cost for three Materialization Strategies

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 total cost computation is given in Table 3 as per the cost computation strategy described in proposed work. This table computes the total cost including the storage cost, maintenance cost & Net benefit interms of number of blocks. Table 2 represents the total cost for all three possibilities, but the proposed method gives the lowest cost than others

Graph 1: Comparison of Query Processing Cost, Maintenance Cost and Storage Cost for three algorithms. The cost is multiple of X10^3

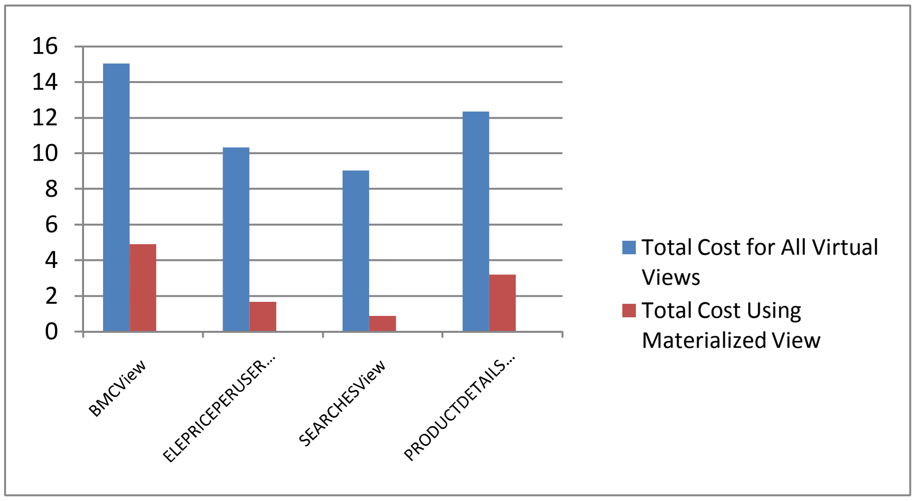


Graph1 shows the comparison of query processing cost, maintenance cost & storage cost for all three approaches including the All materialized view, All virtual views & proposed materialized view & it identifies that the proposed method gives the minimum cost. Most of the algorithms have considered the disk space constraint as a storage cost but now a days it is available in cheap price, therefore we can neglect this parameter while computing the total cost

Table 3: Cost Evaluation of Materialized view in terms of Number of Blocks

Views	**Total (C_{all})**	**Benefit B_i**	**Storage Cost C_{store} (V_i)**	**Maintenance Cost $C_m(V_i)$**	**Net Benefit Net(B_i)**	**Total Cost C_{total}**
BMC View	150456	103458	289	1784	101385	49071
ELEPRICE PER USER View	103290	88930	310	2116	86504	16786
SEARCHES View	90345	82350	156	584	81610	8735
PRODUCT DETAILS View	123504	94356	380	2380	91596	31908

Graph 2: Total Cost Comparison of all Virtual Views & Materialized Views. Cost is multiple of X10^5



The above graph shows that the cost which comes from the materialized is less compared with the all virtual views. We considered the four different views here including the BMCview, ELEPRICEPERUSER View, SEARCHES View, and PRODUCTDETAILSView. The total cost for all these views is less than other. We again compared our proposed algorithm with memetic algorithm (MA) Heuristic algorithm (HA) and Genetic algorithm (GA). Table 4 represents the running time over 10, 20, 40, 60 and 80 queries, respectively. From the experimental results, it can be seen that the running time of the proposed algorithm is fewer than HA and GA in all queries.

Table 4: Comparison of Proposed Method with Other Algorithm Values

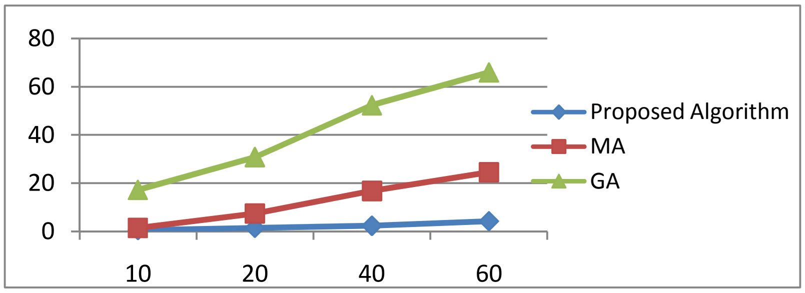
Query	Proposed Algorithm	MA	GA	HA
10	0.5 Min	1.5 Min	17.3 Min	1.2 Hour
20	1.4 Min	7.4 Min	30.9 Min	5.3 Hour
40	2.3 Min	16.8 Min	52.4 Min	10.7 Hour
60	4.2 Min	24.5 Min	1.6 Hour	21.4 Hour
80	6.5 Min	36.3 Min	2.8 Hour	35.6 Hour

We again compared our proposed methodology with CEMS & Optimized CEMS algorithm. Table 5 represents the running time over 10, 20, 40, 60 and 80 queries, respectively. From the experimental results, it is observed that the running time of the proposed algorithm is less than CEMS & Optimized CEMS for all queries.

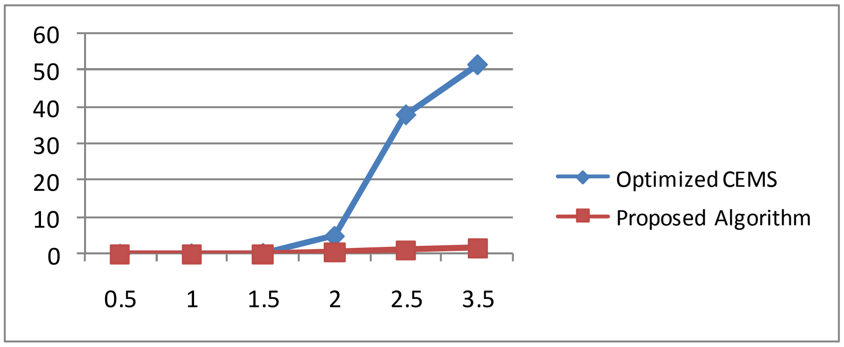
Table 5: Algorithm Comparison based on Database Size and Execution Time.

Database Size (KB)	Proposed Algorithm	CEMS	Optimized CEMS
0.5	***0.078***	0.266	0.188
1	***0.095***	0.297	0.25
1.5	***0.198***	0.39	0.358
2	***0.679***	5.125	5.016
2.5	***0.986***	38.204	38.047
3	***1.589***	51.828	51.688

Graph 4: Execution Time (Sec) vs. Database Size (KB)



Graph 5: Execution Time (Sec) vs. Database Size (KB)



In Graph 4 & 5, the execution time taken by the proposed MV algorithm with memetic algorithm (MA) Heuristic algorithm (HA) and Genetic algorithm (GA). The execution time is given in terms of milliseconds. Here the comparison is also implemented with CEMS &

Optimized CEMS (cost effective approach for Materialized View Selection) on the basis of execution time and it is observed that proposed method requires a minimum time for execution & this minimizes the total cost of query for processing [25,26].

6. CONCLUSION

The materialized view is most beneficial for improving query performance as it stores pre-computed data. But all of the views or queries are not candidates for materialization due to the view maintenance cost. The selection of views to materialize is the important issues in data warehouse. In this article we have outlined a methodology whether the views created for the execution of queries is beneficial or not by considering the various parameters: cost of query, cost of maintenance, net benefit & storage space. We have presented proposed methodology for selecting views to materialize so as to achieve the best combination good query performance. These algorithms are found efficient as compared to other materialized view selection and maintenance strategies. The total cost, composed of different query patterns and frequencies are 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 is 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 which minimizes the total cost of query processing.

REFERENCES:

[1] Gorettiv K.Y. Chan, Qing Li and Ling Feng, "Optimized Design of Materialized Views in a Real-Life Data Warehousing Environment," *International Journal of Information Technology Vol.7, No 1 Sept 2008.*

[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] S Chen and E.A. Rundensteiner, "GPIVOT: Efficient Incremental Maintenance of Complex ROLAP Views," 21st International Conference on Data Engineering (ICDE'05), pp. 552-563, 2005.

[4] A.N.M.B. Rashid and M.S. Islam, "Role of Materialized View Maintenance with PIVOT and UNPIVOT Operators," IEEE International Advance Computing Conference (IACC'09), Patiala, India, pp. 951-955, March 6-7, 2009.

[5] S.R. Valluri, S. Vadapalli, and K. Karlapalem, "View Relevance Driven Materialized View Selection in Data Warehousing Environment," Proceedings of the 13th Australian Database Conference (ADC2002), Melbourne, Australia, vol. 5, pp. 187-196, 2002.

[6] H. Gupta, "Selection of Views to Materialize in a Data Warehouse," Proceedings of ICDT, pp. 98-112, 1997.

[7] H. Gupta, "Selection of Views to Materialize in a Data Warehouse," Proceedings of ICDT, pp. 98-112, 1997.

[8] H. Gupta and I.S. Mumick, “Selection of Views to Materialize under a Maintenance Cost Constraint,” Proceedings of ICDT, pp. 453-473, 1999.

[9] R. Chirkova, A.Y. Halevy, and D. Suciu, “A Formal Perspective on the View Selection Problem,” Proceedings of VLDB, pp. 59–68, 2001.

[10] Gupta, H. & Mumumick, I., “Selection of Views to materialize in a Data warehouse”, *IEEE transactions on Knowledge & Data Engineering, vol: 17, no:1, pp:24-43, 2005.*

[11] 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.*

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

[13] J.Yang, K. Karlapalem and Q. Li, “Algorithms for materialized view design in data warehousing environment,” *Proceedings of Twenty Third Intl. Conf. on Very Large Data Bases, pp.136-145, Aug 1997.*

[14] Gang Gou, Jeffery Xu Yu and Hongjun Lu, “A* Search: An Efficient and Flexible Approach to Materialized View Selection”, *IEEE Trans. on Systems, Man and Cybernetics – Part C: Appl. And Reviews, Vol. 36, No. 3, May 2006.*

[15] 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.*

[16] 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.*

[17] Napster Homepage, http://www.napster.com,

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

[19] B. Babcock, S. Chaudhuri, and G. Das, “Dynamic Sample Selection for Approximate Query Processing,” *Proc. 22nd ACM SIGMOD Int’l Conf. Management of Data (SIGMOD ’03), pp. 539-550, 2003.*

[20] C. Gkantsidis, M. Mihail, and A. Saberi, “Random Walks in Peerto-Peer Networks,” *Proc. IEEE INFOCOM ’04, 2004.*

[21] Benjamin Arai, Gautam Das, Dimitrios Gunopulos, and Vana Kalogeraki, “Efficient Approximate Query Processing in Peer-to-Peer Networks,” *IEEE Trans on Knowledge and Data Engg., Vol. 19, No. 7, Jul 2007.*

[22] C.H. Choi, J. X. Yu and G. Gou, “What difference heuristic make: maintenance cost view selection revisited,” *Proceedings of the third Intl. Conf. on Advances in Web-Age Information Management, Springer-Verlag.pp.313-350, Jan 2002.*

[23] Shantanu Joshi and Christopher Jermaine, “Materialized Sample Views for Database Approximation,” *IEEE Trans on Knowledge and Data Engg., Vol. 20, No. 3, Mar 2008*.

[24] F. Olken, “Random Sampling from Databases,” *PhD dissertation, 1993.*

[25] B.Ashadevi and R.Subramanian, " Optimized Cost Effective Approach for Selection of Materialized views in Data Warehousing", *International Journal of Computer Science and Technology, Vol.9 No.1 ,April 2009.*

[26] Ashadevi and R.Subramanian, "A Cost Effective Approach for Materialized Views Selection in Data Warehousing Environment", *IJCSNS International Journal of Computer Science and Network Security, vol.8 No.10, October2008.*

[27] K.Y.Can, Qing Li ,Lin Fen, "Design and Selection of Materialized Views in a Data warehousing Environment : A Case Study".

[28] A.N.M. Bazlur Rashid and M. S. Islam , "An Incremental View Materialization Approach in ORDBMS" *International Conference on Recent Trends in Information, Telecommunication and Computing 2010*

Author Information

1. Mr. Pravin P.Karde staying at Amravati, Maharstra . He received the Post Graduate Degree (M.E.) in Computer Science & Engineering from S.G.B. Amravati University, Amravati in the year 2006 & pursuing the Ph.D degree in Computer Science & Engineering. Currently he is working as an Assistant Professor & Head in Information Technology Department at H.V.P.M's College of Engineering & Technology, Amravati. His interest is in Selection & Maintenance of Materialized View.

2. Dr. V.M. Thakare staying at Amravati, Maharashtra. He was worked as Assistant Professor for 10 Years at Professor Ram Meghe Institute of Technology & Research, Badnera and P.G.Department of Computer Science, S.G.B. Amravati University, Amravati. Currently he is working as Professor & Head in Computer Science from last 9 years, Faculty of Engineering & Technology, Post Graduate Department of Computer Science, SGB Amravati University, Amravati. He has published 86 papers in various National & International Conferences & 20 papers in various International journals. He is working on various bodies of Universities as a chairman & members. He has guided around 300 more students at M.E / MTech, MCA M.S & M.Phil level. He is a research guide for Ph.D. at S.G.B. Amravati University, Amravati. His interest of research is in Computer Architecture, Artificial Intelligence and Robotics, Database and Data warehousing & Mining.