Cluster Based Framework for Selection of Materialized Views to Reduce the Execution Time and Storage space

Yogeshree D. Choudhari[1]
Asstt. Professor,
Department of Information Technology
K.D.K. College of Engineering
Nagpur, India

Dr. S. K. Shrivastava[2]
Director,
SBITM College of Engineering
Betul, India

ABSTRACT

Data warehouse have been developed to overcome the weakness of traditional databases. A DW is a repository of information collected from multiple, possibly very large, distributed, heterogeneous, autonomous databases and other information sources. It is a set of materialized views defined over remote source relations. It uses multiple materialized views to efficiently process a given set of queries. The DWs are dynamic entities that evolve continuously over time. As time passes, new queries need to be answered by them. Some of these queries can be answered using exclusively the materialized views. When a query is posed, it is evaluated locally, using the materialized views, without accessing the original information sources. It is highly probable that a user will issue a series of similar queries until he or she receives satisfying results. In general new views need to be added to the DW.

Keyword: Materialized view, clustering, Access Frequency, Threshold, %threshold.

1. INTRODUCTION

Quick response time and accuracy are important factors in the success of any database. Materialized views are found useful for fast query processing. A basic requirement for the success of a data warehouse is the ability to provide decision makers with both accurate and timely information as well as fast query response times. In large databases particularly in distributed database, query response time plays vital role as timely access to information and it is the basic requirement of successful business application. 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 View:

A Materialized View (MV) is the pre-calculated (materialized) result of a query. Unlike a simple VIEW the result of a materialized view is stored somewhere, normally in a table. Materialized Views are used when immediate response is needed and the query where the Materialized View bases on would take too long to produce a result. Materialized views are physical structures that improve data access time by pre-computing intermediary query results. Then, end-user queries can be processed efficiently from the data stored within these views and do not need access the original data anymore [1].Materialized Views have to be refreshed for updating it once in a while. It depends on the requirements how often a Materialized View is refreshed and how its actual content is. Basically a Materialized View can be refreshed immediately or deferred; it can be refreshed fully or to a certain point in time.

In this paper, to solve the problem of selection of materialized view, a clustering method is proposed to select the materialized views that reduce the execution time and storage space.

2. Related Work:

A very few research work has been done about selection of materialized view using clustering approach. A significant work about dynamic clustering of Materialized view is done by [1]. It firstly clusters materialized views, and then dynamically adjusts materialized view set and eliminates

the jitter which dynamic materialized view selection algorithm generally has. A heuristics algorithm is developed in [2] that can provide a feasible solution based on individual optimal query plans. It also maps the materialized view design problem as 0-1 integer programming problem. [3] in this paper, a framework for materialized view selection is proposed that exploits a data mining technique (clustering), in order to determine clusters of similar queries. It also proposes 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. An automatic strategy for the selection of XML materialized views that exploit a data mining technique, more precisely the clustering of the query workload is explained in [4]. To validate this strategy, it implemented an XML warehouse modeled along the XCube specifications.

Paper [5] proposes a greedy algorithm BPUS based on the lattice model. And paper [6] discusses the issue of materialized view selection with the B-tree index. Paper [7] proposes PBS algorithm which make the size of materialized view as selection criteria. Paper [8] proposes preprocessor of materialized view selection, which reduces the search space cost and time complexity of static materialized view selection algorithm. These algorithms are based on the known distribution of query, or uniform distribution under the premise, which essentially are static algorithms. However, the query is random in actual OLAP system, so materialized view set which static algorithm generates cannot maximally enhance the query response performance in data warehouse. In order to improve further query response performance in data warehouse, paper [9] proposes dynamic materialized view selection algorithm, FPUS algorithm, which is based on query frequency in unit space. It does not require knowing distribution of query, uniform distribution under the premise neither. However, it dynamically adjusts materialized view according to the collection of query. Paper [10] proposes DCO algorithm. The immediate adjustment strategy of these dynamic selection algorithms improves greatly query response performance.

3. COMPOSITION OF PAPER:

The paper is divided into 4 parts: The first part consist of methodology that describes the problem definition , algorithm and architecture of framework of MV selection, second part is the implementation of the algorithm , third part consist of analytical and experimental results followed by conclusion in the subsequent part.

3.1 IMPLEMENTED METHODOLOGY:

Queries to DW are critical regarding to their complexity and length. They often access millions of tuples and involve joins between relations and aggregations. Materialized views are able to provide the better performance for DW queries. However, these views have maintenance cost, so materialization of all views is not possible. An important challenge of DW environment is materialized view selection because we have to realize the trade-off between performance and view maintenance.

To solve the problem, a clustering method is suggested in which similar queries will be clustered according to their query access frequency to select the materialized views that will reduce the execution time and storage space.

In this, MV based on the access frequency of query and total area required by the posed query is selected .The algorithm selects the required area in such a way that if the MV creation is required for the fired query then only MV is created which will obviously gives the less execution time as compared to DB. The algorithm does not create the MV, if the area required by the posed query doesn't require the selection of MV since the response time of such query by the DB is less than as if the MV would have been created for such queries. Consequently the algorithm selects the very specific MV which really needs to be created to give the less response time compared to DB otherwise queries are directly fired to DB to give the response time less than MV that could have been created. Thus this algorithm basically avoids the creation of MV if it is not necessary for the fired query whose response can directly be obtained from the DB with less execution time as compared to the response time if MV would have been created for such query.

Thus this algorithm focuses on very specific query that satisfy the criteria of certain amount of area selection of table and response time of such query is obviously less than if it is fired to DB directly. Thus this algorithm saves the space by not creating MV for all or usual queries.

3.2. CBFSMV ALGORITHM:

A novel framework is developed for the selection of MV using query clustering. The steps of the algorithm are as below.

I) Generation of random set of records for given tables in database by record generator.

II) Extraction or generation of all possible set of queries resolved by system on above created records.

III) Optimization of above set of queries according to the access frequency of the queries.

IV) Creation of MV according to query access frequency called as Threshold Value and according to Maximum Cluster Area Threshold %.

This **Maximum Cluster Area threshold %** is calculated using following formula.

Cluster Area:
Area of Table (At) = Ctot*Rtot
Area of Cluster (Ac) = Creq*Rreq

Maximum Cluster Area Threshold %(Ra) =Ac/At

Let Threshold_Area_Ratio = Tar

If Ra <= Tar Then
Create View
Else
Ignore

Where, Ctot: Total column of tables
Rtot: Total Records generated for table
Creq: Columns required by query
Rreq: Records required by query

V) According to above criterion of MV creation, 3 types of MV are created as follows.

- **Single query to Multi table MV.**
  In this response of single query is obtained from multiple MV table.

- **Single query to single table MV.**
  In this response of single query is obtained from single MV table.

- **Multiple queries to single table MV.**
  In this response of multiple similar queries will be obtained from single MV table.

VI) After creation of these 3 different types of MV, it will store these MV.

This creation of materialized view depends on the access frequency and threshold% as calculated above. This algorithm will not generate materialized view for the queries whose cluster area will be greater than threshold%. Thus instead of creating the framework it may directly fetch the data from database with less execution time than MV.

3.3 ARCHITECTURE OF FRAMEWORK FOR MV SELECTION:

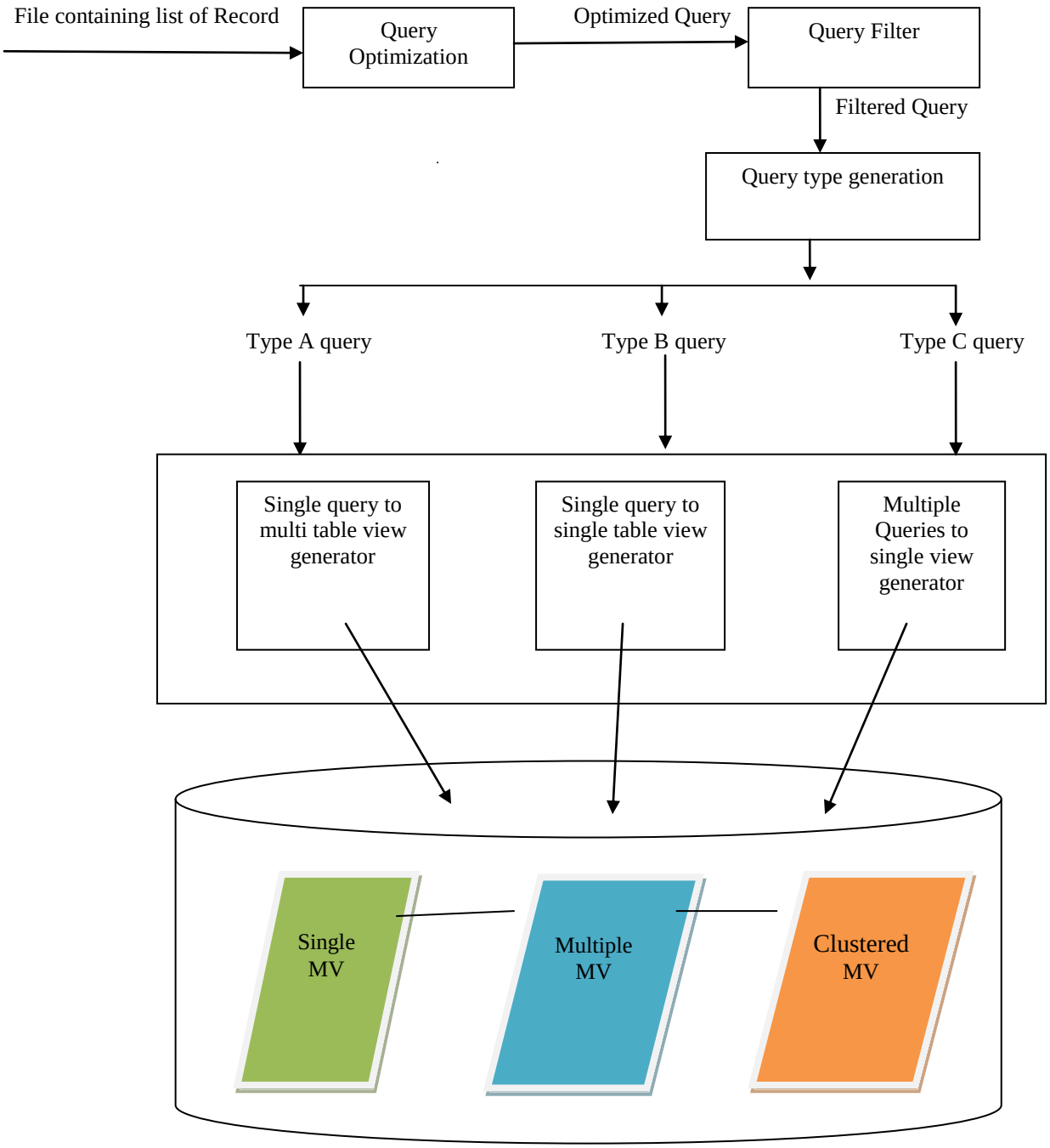

Fig 3.1.: Architecture of Materialized View Selection Framework

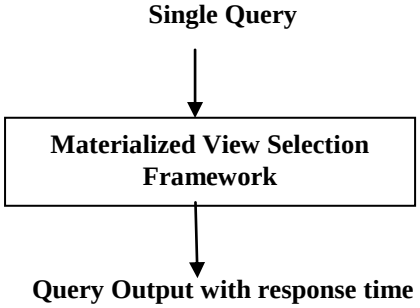

Fig. 3.2. : Query output with Framework with Response Time

4. Analytical and Experimental Results:

The above framework is implemented in NetBeans 6.8. The record generator has created lakhs of random records for given tables.

The given framework is tested for set of random queries.

1] Sample Test Result #1 : For 266 Queries

Record generator has created following no. of records.

1,20,000 records of Custinfo Table.,

1,00,00 records of Emp Table

95,000 records of EmpInfo. Table 132,000 records of StudentInfo. Table

Threshold: 10

%Threshold: 30%

Query1: *select SAge from StudentInfo where (SAge > 20)*
Query2: select Sid from StudentInfo where (SAge > 20)
Query3: select ename, hiredate from EMP where (((hiredate LIKE '1980-02-25') OR (hiredate LIKE '2011-02-25') OR (hiredate LIKE '1981-03-25') OR (hiredate LIKE '1980-02-25')) AND (SAL > 125000)) order by hiredate DESC.

MV(Framework)	TOE	Database(Without Framework)	TOE
Query 1	0	**Query 1**	140
Query 2	0	**Query 2**	109
Query 3	15	**Query 3**	174

Table1: Time of Execution with & without framework.

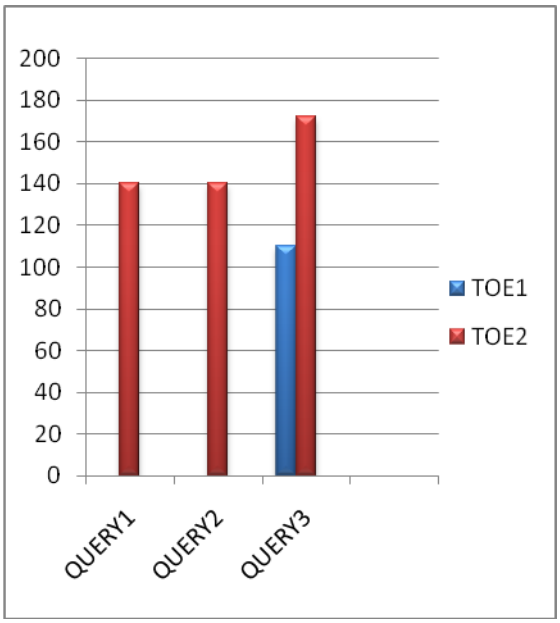

Fig1: Time of Execution of Query1, Query2 and Query3

TOE1: Time of Execution of Queries with framework in nanosecond.
TOE2: Time of Execution of Queries without framework nanosecond.

When query1, query2 and query3 are fired to the DB, MV are created for Query1 , Query2 with TOE1 =0 & Query3=10 ns as compared to TOE2 =94, 140 & 110 ns reply.

2] Sample Test Result #2 : For 532 Queries

Record generator has created following no. of records.

2,00,000 records of Custinfo Table.,Emp Table

2,00,000 records of EmpInfo. Table

2,00,000 records of Studentinfo. Table

Threshold: 20

%Threshold: 30%

MV(Framework)	TOE	Database(Without Framework)	TOE
Query 1	0	**Query 1**	380
Query 2	0	**Query 2**	234
Query 3	10	**Query 3**	266

Table2: Time of Execution of Query1, Query2 and Query3

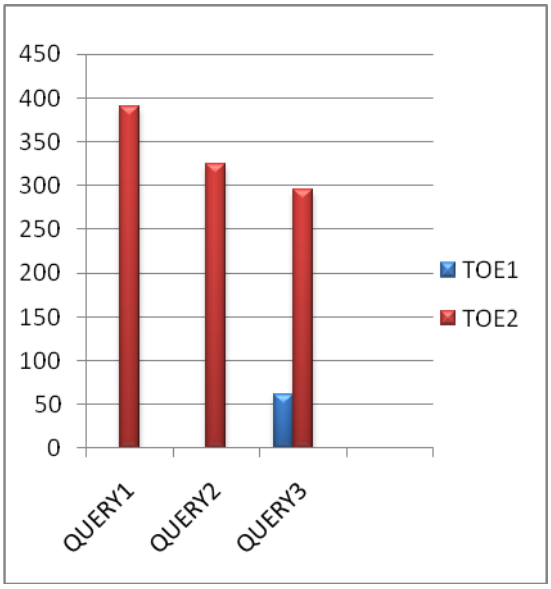

Fig2: Time of Execution of Query1, Query2 and Query3

3] Sample Test Result #3 : For 532 Queries

Record generator has created following no. of records.

3,00,000 records of Custinfo Table, Emp Table

3,00,000 records of EmpInfo. Table

3,00,000 records of Studentinfo. Table

Threshold: 20

%Threshold: 30%

MV(Framework)	TOE	Database(Without Framework)	TOE
Query 1	0	**Query 1**	390
Query 2	0	**Query 2**	324
Query 3	62	**Query 3**	296

Table3: Time of Execution of Query1, Query2 and Query3

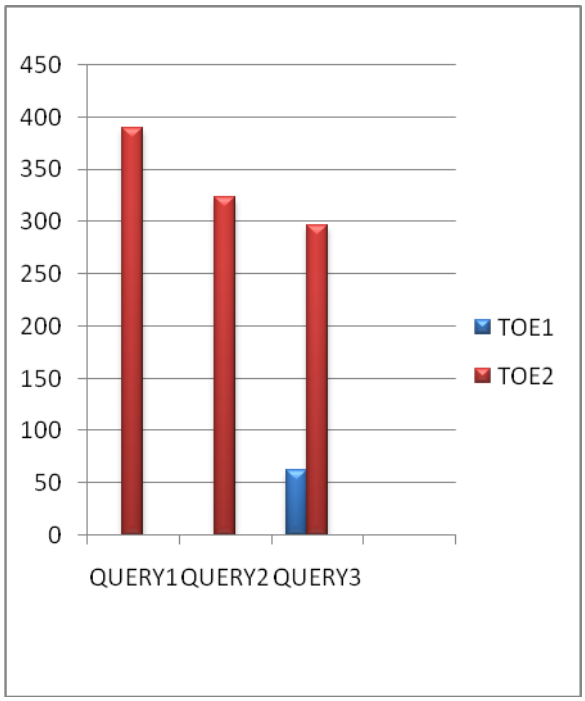

Fig3: Time of Execution of Query1, Query2 and Query3

Above results shows for the queries1 and queries2, response time from MV is 0 .

5. CONCLUSIONS

Thus the paper proposes algorithm for the materialized view design problem, e.g., how to select the set of views to be materialized so that the cost of processing a set of queries and storage space required storing the data for the materialized views is minimized. This approach realizes on analyzing the queries so as to derive common intermediate results which can be shared among the similar queries to reduce the response time and to eliminate the need for creation of same MV for the query. The proposed algorithm for determining a set of materialized views is based on the idea of reusing temporary results from the execution of the global queries. The cost model takes into consideration of both query access frequencies and % threshold.

The work presented here is the first stage research in selection of queries with high access frequencies, clustering them and creation of Materialized Views for the same. These high access frequency queries are further analyzed for required cluster area to create MV.

REFERENCES

[1] An Gong, Weijing Zhao, "Clustering-based Dynamic Materialized View Selection Algorithm**"** *Proceedings of Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, 2008, China,pp391-395.

*2]*Jian Yang,Kamlakar Karlapalem,Quing Li,''Algorithms for materialized view design in data warehousing environment."

3] K. Aouiche, P.Emmanuel Jouve, and J.Darmont ,"Clustering-Based Materialized View Selection in Data Warehouses"*TechnicalReport, University of Lyon 2, 2007.*

4] Hadj Mahboubi, Kamel Aouiche and Jérôme Darmont , "Materialized View Selection by Query Clustering in XML Data Warehouses" *Fourth International Conference on Computer Science and Information Technology-Jor dan.*

[5] Harinarayan V, Rajaraman A, Ullman J D, "Implementing Data Cubes Efficiently", *Proceedings of ACM SIGMOD International Conference Management of Data*, 1996, pp. 205-216.

[6] Gupta H, Harinarayan V, Rajaraman A, et al, "Index Selection for OLAP", *Proceeding of International Conference on Data Engineering*, 1997, pp. 208-219.

[7] Shukla A, Deshpande P M, Naughton J F, "Materialized View Selection or Multidimensional Datasets", *Proceedings of the 24th VLDB Conference*, 1998, pp. 488-499.

[8] Zhang Bai Li, Sun Zhi Hui, and Sun Xiang, "Preprocessor of Materialized Views Selection", *Journal of Computer Research and development*, 2004, pp. 1645-1651.

[9] Tan Hong xing, Zhou Long xiang, "Dynamic Selection of Materialized Views of Multi-Dimensional Data", *Journal of Software*, 2002, pp. 1090-1096.

[10] Zhang BL, Sun ZH, Zhou XY, et al, "A Dynamic Cache Optimized Algorithm of Static Materialized Views", *Journal of Software*, 2006, pp. 1213-1221.

[11] Mistry H, Roy P, Sudarshan S, et al, "Materialized view selection and maintenance using multi-query optimization", *Proceedings of SIGMOD'01*, 2001, pp. 307-318.

[12] Shukla A, Deshpande PM, Naughton JF, "Materialized view selection for multidimensional datasets", *Proceedings of the 24th International Conference on VLDB*, Morgan Kaufmann Publishers, San Francisco, 1996, pp. 51.