An Architectural Framework for Constructing Materialized Views in a Data Warehouse

T. V. Vijay Kumar and Kalyani Devi

Abstract—**Data warehouse stores data accumulated over a period of time from disparate data sources for providing answers to analytical queries. These queries, which are long and complex in nature, have high query response time when processed against a large data warehouse. This response time can be reduced by constructing materialized views and storing them in a data warehouse. These views need to contain relevant and required information for answering future queries, so that future queries can be answered in a reduced response time to make decision making more efficient. A Materialized Views Construction Framework (MVCF), presented in this paper, lays down a strategy for constructing materialized views from previously posed queries on the data warehouse. The objective of MVCF is to enable construction of materialized views that are subject-specific and contain frequently accessed information that are capable of providing answers to future queries. This in turn would facilitate decision making.**

Index Terms—**Data warehouse, materialized view.**

I. Introduction

Contemporary business organizations are eager to exploit data available in disparate data sources spread across the globe. This data, being valuable for decision making, needs to be organized in a manner so that decisions can be made quicker. Most organizations store this data continuously over a period of time in a data warehouse on which analytical queries can be posed for decision making [1]. These analytical queries are long and complex in nature and, when posed against a large data warehouse consume a lot of time for processing. Analytical processing being exploratory in nature, results in further delay in decision making. Though effective solutions exist to represent data for analytical queries, the problem of high query response time still needs to be addressed adequately [2]. This problem has been addressed to some extent using query optimization techniques [3], [4] and indexing techniques [5] but these do not scale for large amounts of data in the data warehouse. An alternative way to address this problem is by storing materialized views in a data warehouse [6]. These views contain pre-computed aggregate, or summarized, information and are stored in a data warehouse for the purpose of answering analytical or decision making queries in a reduced response time. It eliminates the overheads associated with expensive joins and aggregations [7]. Materialized views have several issues associated with them like view selection, view maintenance, view evolution and answering queries using views [8]. This paper focuses on the view selection issue [9]. View selection is concerned with selecting an appropriate set of views that improves the query response time and fits within the available storage space for materialization [9], [10]. It aims to achieve a trade-off between the query response time and the available storage space [11] in order to ensure optimal performance of the system [12], [13].

Though materializing all possible views may reduce the query response time significantly, it is not feasible to store views in the available space for materialization for higher dimensional data sets due to the number of views being exponential with respect to the number of dimensions. Thus, there is a need to select a subset of views from amongst all possible views. This selection cannot be done arbitrarily as it may result in materialized views having data that may not be used for answering queries and relevant data for answering queries may not be in the selected materialized views. Further, optimal view selection is shown to be an NP-Complete problem [12]. Several view selection approaches exist in literature, most of which are heuristic based or empirical based. Several heuristic based approaches exist for materialized view selection with most of them being either greedy based [10], [12], [14]-[18] or evolutionary based [19], [20], [21]. Empirically, materialized views can be selected by monitoring the queries, submitted by users, and assessing them on factors like frequency or size of data in order to materialize appropriate views [11], [22]. This paper presents a framework that lays out a strategy for empirical view selection.

Most existing empirical based approaches for view selection are workload driven. Their basis is that past query workloads are indicative of the queries likely to be posed in future and materialized views constructed using them have a greater likelihood of answering future queries. Large numbers of queries may be there in the query workload. Considering all queries for constructing materialized views may not be appropriate as some of these may not be relevant and useful, and may not be accessed frequently. They may increase the cost without contributing much towards answering future queries. Those queries that maximize the profit with respect to answering future queries should be materialized. The selection of such materialized views is a complex problem. This problem is addressed by the Materialized View Construction Framework (MVCF), presented in this paper, which lays down a strategy for constructing materialized views using previously posed queries on a data warehouse. The framework objective is to enable construction of materialized views that have high

Manuscript received October 13, 2012; revised February 16, 2013.

T. V. Vijay Kumar is with School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi-110067 (e-mail: tvvijaykumar@hotmail.com).

Kalyani Devi is with Business Solutions Department in Siemens Information Systems Limited, Gurgaon, Haryana-122001 (e-mail: devi_kalyani14@yahoo.co.in).

likelihood of answering future queries. This paper is a revised version of [23].

The paper is organized as follows: The architectural framework MVCF is discussed in Section II. Section III briefly discusses a system MVCS based on the framework MVCF. Experimental Results are discussed in Section IV followed by conclusion in section V.

II. MVCF

The Materialized View Construction Framework MVCF [23] describes a framework for constructing materialized views using previously posed queries on the data warehouse. This framework assumes that the queries, likely to be posed on the data warehouse in future, shall be similar to the queries posed on the data warehouse in the past and thus these previously posed queries can be used for constructing materialized views over a data warehouse. MVCF assumes that a Query Log comprising of these previously posed queries is maintained. This Query Log would then be used to construct materialized views in a data warehouse. MVCF aims to construct materialized views that are subject or domain specific as most queries in the data warehouse are subject specific. MVCF ensures this by grouping closely related queries in a Query Log into clusters or groups of queries, where each such group specifies a subject area or domain in the data warehouse. MVCF then describes a mechanism to prune out infrequent and non-optimal queries from each domain. MVCF first identifies frequent queries in each domain and follows it up by selection of optimal queries from amongst these. A merging strategy is specified for merging these optimal queries to construct materialized views for the respective domain. This construction of materialized views using MVCF is described by four phases namely Domain Creation, Frequent Query Identification, Optimal Queries Selection and Optimal Queries Merging. This framework MVCF [23] is given in Fig. 1.

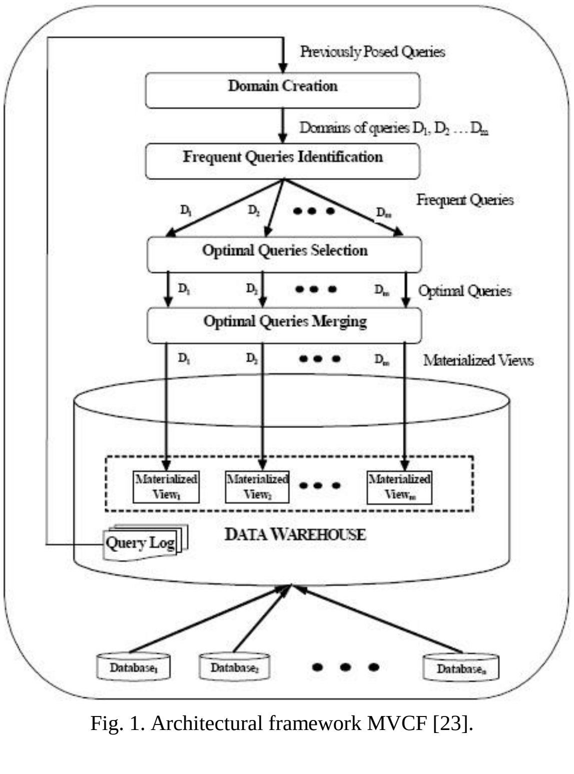


Fig. 1. Architectural framework MVCF [23].

A. Domain Creation

A data warehouse stores subject specific data [1] with most queries posed on the data warehouse, being subject specific. MVCF aims to construct subject specific materialized views so that any future query posed on the data warehouse can be answered by fewer materialized views. Thus the first phase of MVCF is to create subject areas, or domains, comprising closely related queries with respect to the data accessed by them. MVCF uses the queries in the Query Log and groups them based on their similarity in terms of the data accessed by them. The similarity between any two queries can be computed using similarity measures like Jaccard, Dice, Overlap, Cosine etc [24]. These can be computed as given below:

Jaccard's Coefficient [24], [25]: This similarity is defined as the proportion of number of relations in common accessed by a pair of queries. Let R and S be the set of relations accessed by the pair of queries Q_i and Q_j, then the similarity between queries Q_i and Q_j, i.e. SIM (Q_i, Q_j), based on Jaccard's coefficient is given as

$$Sim\left(Q_i,Q_j\right)=\frac{\left|R\cap S\right|}{\left|R\cup S\right|}$$

Cosine Similarity [26], [27]: Let $Q_1, Q_2, \ldots, Q_n$ be a set of queries, where each query Qj can be represented in terms of relation and weight vector as

$$Q_j = \{<R_1, w_{1Qj}>, <R_2, w_{2Qj}>, \ldots, <R_m, w_{mQj}>\}$$

where $R_1, R_2, \ldots R_m$ are the relations used in the query Q_j and $w_{1Qj}, w_{2Qj}, \ldots, w_{mQj}$ are the weight vectors, which are defined as

$$w_{iQj} = r_{fiQj} \times i_{qfi}$$

where rf_{iQj} is the number of occurrence of relation R_i in Q_j, i.e. the relation frequency in query Q_j and $iqf_i = \log(n/qfi)$, where n is the total number of queries and qf_i is the number of queries containing relation R_i i.e. the Inverse Query Frequency.

The similarity between two queries Q_i and Q_j using the cosine similarity measure is

$$sim(Q_i,Q_j)=\frac{\sum_{i=1}^{k} cw_{iQ_i}\times cw_{iQ_j}}{\sqrt{\sum_{i=1}^{k} cw_{iQ_i}^{\ 2}}\times\sqrt{\sum_{i=1}^{k} cw_{iQ_j}^{\ 2}}}$$

where cw_{iQi} refers to the weight of the i^{th} common relation of $C_{ij} = \{r: r \in Qi \cap Qj\}$ in query Qi

Hybrid Similarity [26], [28]: This measure is defined as

$$SimHybrid\left(Q_i,Q_j\right)=\alpha\times\frac{\left|Q_i\cap Q_j\right|}{Max\left(\left|Q_i\right|,\left|Q_j\right|\right)} + \beta\times\frac{\sum_{i=1}^{k} CW_iQ_i\times CW_iQ_j}{\sqrt{\sum_{i=1}^{k} W_1Q_1^{\ 2}}\times\sqrt{\sum_{i=1}^{k} W_1Q_1^{\ 2}}}$$

where $|Qi \cap Qj|$ is the number of common attributes in queries *Qi* and *Qj* respectively.

OVERLAP Coefficient [29], [30]: The Similarity between a pair of queries Q_i and Q_j, i.e. Sim(Q_i, Q_j), based on the Overlap Coefficient measure, is given by

$$Sim(Q_i\,,\,Q_j) = \frac{|\,R(Q_i) \cap R(Q_j)\,|}{Min(|\,R(Q_i)\,|,|\,R(Q_j)\,|)}$$

where $R(Q_i)$ and $R(Q_j)$ are the relations accessed by queries Q_i and Q_j respectively.

DICE Coefficient [31], [32]: The Similarity between a pair of queries Q_i and Q_j, i.e. Sim(Q_i, Q_j), based on Dice Coefficient measure, is given by

$$Sim(Q_i\,,\,Q_j) = \frac{2\,|\,R(Q_i) \cap R(Q_j)\,|}{|\,R(Q_i)\,| + |\,R(Q_j)\,|}$$

where $R(Q_i)$ and $R(Q_j)$ are the relations accessed by queries Q_i and Q_j respectively.

The above similarity measures can be used to construct the query similarity matrix. This similarity matrix can be further used to group closely related queries into cluster queries. These cluster queries can be formed using clustering techniques [33] like Agglomerative hierarchical clustering [25], DBSCAN [27], KMEANS [28], OPTICS [30], Nearest Neighbor [32]. Each of the clusters created using the above techniques would contain queries that are strongly related to each other with respect to the data accessed by them. These clusters of queries specify the various domains in the data warehouse. A materialized view would then be constructed for each such domain.

B. Frequent Queries Identification

MVCF defines a framework for constructing a materialized view for each domain using the queries therein. It is possible that all queries in a domain may not be of equal importance, as they may be accessing data that was rarely accessed in the past. These queries may not be appropriate for constructing materialized views and thus need to be pruned out. On the other hand, queries in a domain accessing similar data are indicative of the data that is frequently accessed and therefore need to be identified for each domain. These queries, referred to as frequent queries, are identified next in the framework MVCF. The framework suggests that these frequent queries can be identified using the association rule mining techniques. A query in a domain is a transaction with relations in it being the item sets or relation sets. So the number of transactions is the number of queries in each domain. Using these transactions, frequent relation sets are identified with a pre-specified query support threshold. These frequent relation sets can be identified using the association rule mining techniques [34] like Apriori [25], FP Tree [27], Pincer Search [28], DHP [30], DIC [32]. These identified frequent relation sets consist of relations that have been frequently accessed by previously posed queries and thus have a high likelihood of being accessed in future. MVCF uses these frequent relation sets to identify frequent queries in each domain. The queries that contain any one of the frequent relation sets in a domain are identified as frequent queries. These queries provide indicators to frequently accessed data and therefore can be considered appropriate for the construction of materialized views.

C. Optimal Queries Selection

A materialized view aims to improve the query response time of analytical queries while conforming to the available space for materialization. It needs to be ensured that this available space is optimally utilized with respect to the materialized views stored in it. Thus, a materialized view should only be constructed using frequent queries that are profitable with respect to answering future queries. This necessitates constructing materialized views using frequent queries that maximize the profit in terms of their containing information that can provide answers to most future user queries. Thus frequent queries, which contribute to maximizing the profit of materialization, are selected from amongst all the frequent queries in a domain. The queries that contribute more towards cost, without contributing much towards the profit, are pruned out. On the other hand, frequent queries that are profitable and less costly are selected and used for constructing materialized views. The framework suggests that these profitable queries, referred to as optimal queries, can be selected by formulating the selection problem as a zero-one integer programming problem [35], [36] with the frequent query being the decision making variable taking either value 1 or 0 depending on whether it is profitable or not. The objective is to select frequent queries that maximize the profit of materialization in respect of containing information that can provide answers to maximum numbers of queries while minimizing the cost, due to various resource constraints like size, CPU usage, memory utilization, running time etc., associated with them. The frequent queries having value 1, termed as optimal queries, are selected for constructing materialized views for the corresponding domain. Greedy algorithms or evolutionary algorithms can also be used to select such optimal queries.

D. Optimal Queries Merging

The framework MVCF considers the use of optimal queries in a domain to construct a materialized view for it. The domain may contain one or more optimal queries. If there is a single optimal query then the view over it would be materialized. In case, multiple optimal queries in a domain, then either one materialized view is constructed for each optimal query or a single materialized view is constructed by merging all optimal queries. In the former case, the number of materialized view constructed in a domain would be same as the number of optimal queries in that domain. The frame work MVCF suggests the latter approach for constructing materialized views, as the optimal queries in a domain are closely related and in such scenario constructing materialized views for each optimal query would lead to sub-optimal solution.

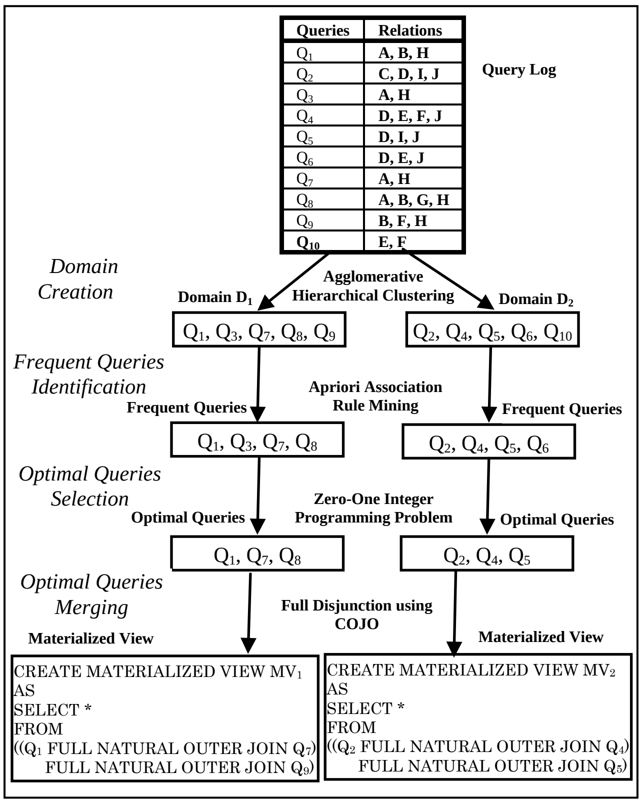
Merging of optimal queries in a domain should ensure that no information should be lost while merging. This would necessitate maximal integration of information produced from each optimal query. That is, merging of optimal queries should preserve information in them and the resultant single materialized view should be meaningful or a full disjunction [37]. The framework MVCF suggests an existing natural outer join ordering strategy that defines the order in which optimal queries should be merged so that the resultant materialized view is a full disjunction. The natural outer join ordering can be computed using algorithms like SOJO [38] or COJO [39]. The optimal queries can then be merged in the natural outer join order defined by these algorithms. The resultant merged query is the maximal integration of information contained in the optimal queries. A materialized view is then constructed using the merged query.

As prescribed by MVCF, a single materialized view is constructed for each domain. Thus, the number of materialized views constructed is the same as the number of domains in the data warehouse. These materialized views are stored in the data warehouse and future queries are first evaluated against these relatively smaller sized materialized views instead of the large data warehouse. As a result, search space and volume of data to be processed would be less. This would lead to considerable reduction in the query response time and aids in decision making.

The materialized views, so constructed, should be periodically assessed and recomputed to reflect, and satisfy, the changing trends in user querying. MVCF suggests that the materialized views are recomputed in the same manner as they are constructed using a recent set of previously posed queries in the QueryLog.

III. MVCS

A Materialized View Construction System (MVCS), based on the framework MVCF, is presented in [40]. MVCS uses previously posed queries on the data warehouse to construct materialized views. First, it computes the similarity between previously posed queries using Jaccard's Coefficient [24]. The computed similarities are then used to construct the similarity matrix. MVCS then groups closely related previously posed queries into clusters of queries using the Agglomerative Heirarchical clustering technique [33]. Each such cluster specifies a subject-specific domain in the data warehouse. MVCS then identifies frequent relations sets using the queries in each domain using the Apriori association rule mining technique [41]. The queries containing any of these frequent relation sets are identified as frequent queries in the domain. MVCS formulates the selection of optimal queries, from among the frequent queries, in a domain as a zero-one integer programming problem [36]. The frequent queries in the solution having value 1 are selected as optimal queries. These optimal queries in a domain are then merged [42]. A materialized view is then created using this merged query. MVCS constructs a single materialized view for each domain. An illustrative example showing the construction of materialized views using the queries in Query Log is shown in Fig. 2. In Fig. 2, the SQL queries Q_1, Q_2, … Q_{10} in the Query Log are used by MVCS to construct materialized views. The relations in the FROM clause of these queries are grouped together, using agglomerative hierarchical clustering technique, into two clusters of queries specifying the two domains D_1= {Q_1, Q_3, Q_7, Q_8, Q_9} and D_2 = {Q_2, Q_4, Q_5, Q_6, Q_{10}} in the data warehouse. MVCS then identifies the frequent queries in each domain using Apriori association rule mining technique. The frequent queries identified in domain D_1 are {Q_1, Q_3, Q_7, Q_8} and the frequent queries identified in domain D_2 are {Q_2, Q_4, Q_5, Q_6}. The optimal queries selection in each domain is formulated as a zero-one integer programming problem. Those frequent queries having value 1 in the solution are selected as optimal queries in the respective domain. The optimal queries selected in domain D_1 are {Q_1, Q_7, Q_8} and in domain D_2 are {Q_2, Q_4, Q_5}. These optimal queries, in each domain, are then merged in natural join order that leads to full disjunction, as defined by COJO [39]. Materialized views MV_1 and MV_2 are thereafter constructed over the merged query in domains D_1 and D_2 respectively.



Queries	Relations
Q_1	A, B, H
Q_2	C, D, I, J
Q_3	A, H
Q_4	D, E, F, J
Q_5	D, I, J
Q_6	D, E, J
Q_7	A, H
Q_8	A, B, G, H
Q_9	B, F, H
Q_{10}	E, F

```
CREATE MATERIALIZED VIEW MV1
AS
SELECT *
FROM
((Q1 FULL NATURAL OUTER JOIN Q7)
    FULL NATURAL OUTER JOIN Q9)
```

```
CREATE MATERIALIZED VIEW MV2
AS
SELECT *
FROM
((Q2 FULL NATURAL OUTER JOIN Q4)
    FULL NATURAL OUTER JOIN Q5)
```

Fig. 2. Materialized view construction using MVCS.

IV. Experimental Results

The experiments were carried out by using a data warehouse of size 2 GB stored in RDBMS ORACLE 9i installed on a Pentium 2.3 GHz machine with 512 MB RAM and 80 GB Hard Disk. MVCS [40], based on the framework MVCF [23], considered 200 previously posed queries in the QueryLog. It used 100 queries to construct the materialized views and the remaining 100 queries were used to validate the performance of MVCS.

Firstly, the query response time due to materialized views were evaluated and plotted as a bar graph, shown in Fig. 3. The bar graph showed how the query response time varied, with increase in the number of queries, when materialized views were used and when they were not used.

It can be noted from the graph that the use of materialized views has led to an improvement in the query response time.

Thus, it can be inferred that MVCS is able to construct materialized views that result in efficient query processing and thereby aid in decision-making.

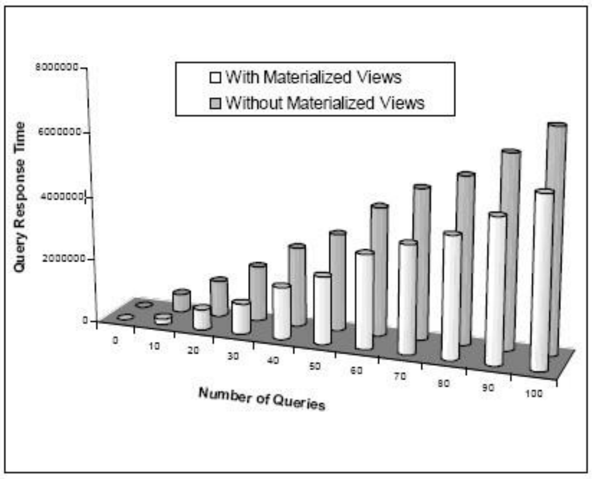


Fig. 3. Query response time with & without materialized views.

Next, in order to observe the extent to which queries are answered by the materialized views i.e. the number of queries that are answered 100%, 75%, 50%, 25% and 0%, a bar graph, shown in Fig. 4, is plotted between the number of queries answered by materialized views and the number of queries processed against the data warehouse. It is observed from the bar graph that materialized views are able to provide some answers to most queries while a good number of queries are almost completely (above 50%) answered by them. Out of the 100 queries posed, materialized views provide answers to 80 queries. Of these 17, 13, 26 and 24 queries are answered 100%, 75%, 50% and 25% respectively. The materialized views are unable to provide answers to 20 queries. Thus, it can be said that materialized views are able to answer reasonably large number of queries.

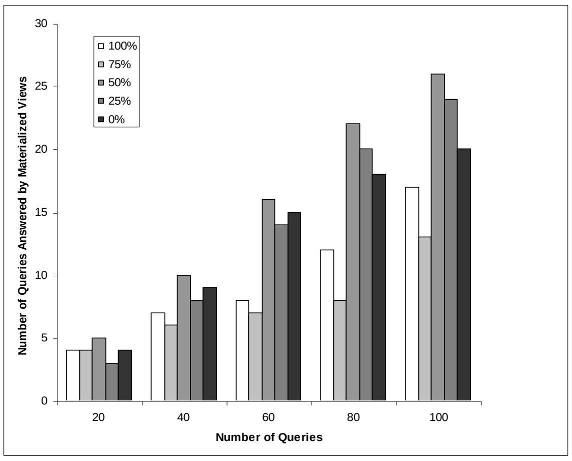


Fig. 4. Queries answered in percentages *vs*. number of queries.

Further, in order to ascertain the query response time with increase in the storage space for materialization, a graph of query response time vs. available storage space is plotted and is shown in Fig. 5. The graph shows that, as the available space increases, there is a significant improvement in the query response time. This may be due to the fact that as the available storage space increases, more views can be materialized leading to greater number of queries being answered. This in turn leads to a reduction in the query response time.

It can be concluded from the above graphs that the materialized views constructed by MVCS, based on the framework MVCF, are capable of providing answers to most future queries. This in turn would facilitate decision-making.

V. Conclusion

In this paper, an architectural framework MVCF that lays down a phase wise strategy for constructing materialized views in a data warehouse is presented. This framework suggests that previously posed queries, which are indicative of the queries likely to be posed in future, are appropriate for constructing materialized views. As per MVCF, these queries are grouped into clusters of queries, based on their similarity with respect to the data accessed by them. Each such cluster specifies a domain in the data warehouse. Then frequent queries are identified in each domain. This is followed up by selection of optimal queries from amongst these frequent queries. These optimal queries in each domain are merged and materialized view is constructed over them. MVCF constructs domain specific materialized views. Since most queries posed on the data warehouse are domain specific, fewer views would be required to answer them. MVCF aims to provide a mechanism for selecting optimal queries with respect to providing answers to user queries. Materialized views constructed using these optimal queries, enables the queries to be answered using them without requiring their processing against the data warehouse. As a result, the query response time would be reduced and decision making would become more efficient.

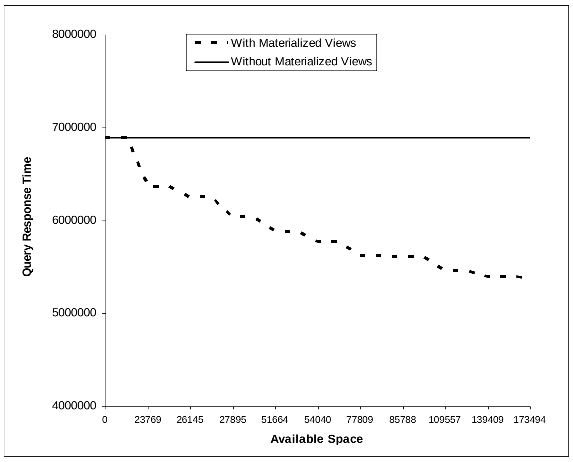


Fig. 5. Query response time *vs*. available space.

The framework MVCF was validated by analyzing the performance of the system MVCS [40] based on it. Experiments were performed on MVCS using previously posed queries on the data warehouse. Results of the experiments show that MVCS is able to construct materialized views that are able to reduce the query response time by providing answers to a reasonably greater number of queries. This shows that the strategy laid by the framework MVCF for constructing materialized views can facilitate the decision making process.

REFERENCES

[1] W. H. Inmon, *Building the Data Warehouse*, Third Edition, Wiley Dreamtech, 2003

[2] B. Shah, K. Ramachandran, and V. Raghavan, "A Hybrid Approach for Data Warehouse View Selection," *International Journal of Data Warehousing and Mining*, vol. 2, no. 2, pp. 1–37, 2006

[3] S. Chaudhuri and K. Shim, "Including Groupby in Query Optimization," in *Proc. the International Conference on Very Large Database Systems*, 1994

[4] A. Gupta, V. Harinarayan, and D. Quass, "Generalized Projections: A Powerful Approach to Aggregation," in *Proc. 21st VLDB conference*, pp. 11-15, 1995

[5] P. O'Neil and G. Graefe, "Multi-Table joins through Bitmapped Join Indices," *SIGMOD Record*, vol. 24, no. 3, pp. 8-11, 1995

[6] N. Roussopoulos, "Materialized Views and Data Warehouse," 4th Workshop KRDB-97, Athens, Greece; August 1997

[7] T. S. Jung, M. S. Ahn, and W. S. Cho, *An Efficient OLAP Query Processing Technique Using Measure Attribute Indexes*, pp. 218-228, 2004.

[8] M. Mohania, S. Samtani, J. Roddick, and Y. Kambayashi, "Advances and Research Directions in Data Warehousing Technology," *Australian Journal of Information Systems*, 1998

[9] R. Chirkova, A. Y. Halevy, and D. Suciu, "A Formal Perspective on the View Selection Problem," in *Proc.* VLDB, pp 59-68, 2001

[10] H. Gupta, V. Harinarayan, V. Rajaraman and J. Ullman, "Index Selection for OLAP," in *Proc. the 13th International Conference on Data Engineering, ICDE 97*, Birmingham, UK, 1997

[11] M. Teschke and A. Ulbrich, "Using Materialized Views to Speed Up Data Warehousing," Technical Report, IMMD 6, Universität Erlangen-Nümberg, 1997

[12] V. Harinarayan, A. Rajaraman and J. D. Ullman, "Implementing Data Cubes Efficiently," ACM SIGMOD, Montreal, pp. 205-216, 1996

[13] D. Theodoratos and T. Sellis, "Data Warehouse Configuration," in *Proc. VLDB*, pp. 126-135, Athens, Greece, 1997

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

[15] S. Valluri, S. Vadapalli, and K. Karlapalem, "View Relevance Driven Materrialized View Selection in Data Warehousing Environment," *Australian Computer Science Communications*, vol. 24, no. 2, pp. 187-196, 2002

[16] T. V. Vijay Kumar and A. Ghoshal, "A Reduced Lattice Greedy Algorithm for Materialized Views Selection," *Communications in Computer and Information Science (CCIS)*, vol. 31, pp. 6-18, 2009

[17] T. V. Vijay Kumar and A. Ghoshal, "Greedy Selection of Materialized Views," *International Journal of Computer and Communication Technology (IJCCT)*, vol. 1, pp. 47-58, 2009

[18] T. V. Vijay Kumar, M. Haider, and S. Kumar, "Proposing Candidate Views for Materialization," *Communications in Computer and Information Science (CCIS)*, vol. 54, pp. 89-98, 2010

[19] J. T. Horng, Y. J. Chang, B. J. Liu, and C.Y. Kao, "Materialized View Selection Using Genetic Algorithms in a Data warehouse System," in *Proc. the 1999 congress on Evolutionary Computation*, Washington D. C., vol. 3, 1999

[20] M. Lawrence, "Multiobjective Genetic Algorithms for Materialized View Selection in OLAP Data Warehouses," in *Proc. GECCO'06*, July 8-12, Seattle Washington, USA, 2006

[21] C. Zhang, X. Yao, and J. Yang, "An Evolutionary Approach to Materialized Views Selection in a Data Warehouse Environment," *IEEE Transactions on Systems, Man and Cybernetics*, pp. 282-294 2001

[22] W. Lehner, T. Ruf, and M. Teschke, "Improving Query Response Time in Scientific Databases Using DataAggregation, In proceedings of 7th International Conference and Workshop on Database and Expert Systems Applications, DEXA 96, Zurich, 1996

[23] T. V. Vijay Kumar and K. Devi, A Materialized View Construction Framework for a Data Warehouse, in *Proc. the 2nd International Conference on Information and Multimedia Technology (ICIMT-2010)*, vol. 3, pp. 165-169, 2010

[24] B. Merkines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme, "Evaluating Similarity Measure for Emergent Semantics of Social Tagiing," in *Proc. 18th International World Wide Web Conference (WWW 2009)*, pp. 641-650, 2009

[25] T. V. Vijay Kumar and K. Devi, "Frequent Queries Identification for Constructing Materialized Views," in *Proc. the 3rd International Conference on Electronics Computer Technology(ICECT-2011)*, vol. 6, pp. 177-181, 2011

[26] L. Fu, D. H. Goh, and S. Foo, "Query Clustering using a Hybrid Query Similarity Measure," *WSEAS Transactions on Computers*, vol. 3, no. 3, pp. 700-705. 2004

[27] T. V. Vijay Kumar, A. Goel, and N. Jain, "Mining Information for Constructing Materialised Views," *International Journal of Information and Communication Technology*, vol. 2, no. 4, pp. 386-405, 2010

[28] T. V. Vijay Kumar and N. Jain, "Selection of Frequent Queries for Constructing Materialized Views in Data Warehouse," *The IUP Journal of Systems Management*, vol. 8, no. 2, pp. 46-64, May 2010

[29] T. E. Clemons and E. L. Bradley Jr., "A nonparametric measure of the overlapping coefficient," *Journal of Computational Statistics & Data Analysis*, vol. 34, issue 1, July 28, 2000

[30] T. V. Vijay Kumar, A. Singh, and G. Dubey, "Mining Queries for Constructing Materialized Views in a Data Warehouse," *Advances in Intelligent and Soft Computing*, vol. 167, pp. 149-159, 2012

[31] L. R. Dice, "Measures of the Amount of Ecologic Association Between Species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945

[32] T. V. Vijay Kumar, G. Dubey, and A. Singh, "Frequent Queries Selection for View Materialization," *Advances in Intelligent Systems and Computing*, vol. 177, pp. 521-530, 2012

[33] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264-323 September, 1999

[34] A. Ceglar and J. F. Roddick, "Association Mining," *ACM Computing Surveys*, vol. 38, no. 2, pp. 1-42, July 2006

[35] H. A. Taha, *Operations Research: An Introduction*, 7th Edition, Pearson Education, 2003

[36] T. V. Vijay Kumar and K. Devi, "Optimal Queries Selection for Constructing Materialized Views," presented at the 40th Annual Convention of Operational Research Society of India, (A Golden Jubilee Celebration of ORSI), December 4–6, 2007

[37] C. A. Galindo-Legaria, "Outerjoins as Disjunctions" in *Proc. ACM-SIGMOD International Conference on Managament of Data*, Minneapolis, USA, pp. 348-358, 1994

[38] A. Rajaraman and J. D. Ullman, "Integrating Information by Outerjoins and Full Disjunctions," *PODS*, pp. 238-248, 1996

[39] T. V. Vijay Kumar, A. Shridhar, and A. Ghoshal, "Computing Full Disjunction using COJO," *Information Technology and Management*, vol. 10, no. 1, pp. 3-20, March 2009

[40] K. Devi and T. V. Vijay Kumar, "Materialized View Selection in Data Warehouse," *PCTEJCS*, pp. 33-41, Jan-June 2007

[41] R. Agarwal and R. Srikant, "Fast Algorithms for Mining Association rules," in *Proc. International Conference on Very Large Database, Chile*, September, pp 487-499, 1994

[42] T. V. Vijay Kumar and K. Devi, "Materialized View Construction in Data Warehouse for Decision Making," I*nternational Journal of Business Information Systems (IJBIS)*, vol. 11, no. 4, pp.379–396, 2012

T. V. Vijay Kumar received his PhD in the area of databases from School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India after completing his MPhil and MSc in Operational Research and BSc (hons.) in Mathematics from the University of Delhi, Delhi, India. His research interests include databases, data warehousing, data mining and soft computing.

Kalyani Devi received her MTech in Computer Science and Technology from School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India after completing her MCA and BSc (hons.) in Computer Science from Utkal University, Orissa, India. Her areas of interest include databases and data warehousing.