A Comprehensive Analysis of Materialized Views in a Data Warehouse Environment

Garima Thakur
M.Tech (IT), Department of IT
USIT, Guru Gobind Singh Indraprastha University
Delhi, India
thakur_garima_27@yahoo.co.in

Anjana Gosain
Associate Professor, Department of IT
USIT, Guru Gobind Singh Indraprastha University
Delhi, India
anjana_gosain@hotmail.com

***Abstract*— Data in a warehouse can be perceived as a collection of materialized views that are generated as per the user requirements specified in the queries being generated against the information contained in the warehouse. User requirements and constraints frequently change over time, which may evolve data and view definitions stored in a data warehouse dynamically. The current requirements are modified and some novel and innovative requirements are added in order to deal with the latest business scenarios. In fact, data preserved in a warehouse along with these materialized views must also be updated and maintained so that they can deal with the changes in data sources as well as the requirements stated by the users. Selection and maintenance of these views is one of the vital tasks in a data warehousing environment in order to provide optimal efficiency by reducing the query response time, query processing and maintenance costs as well. Another major issue related to materialized views is that whether these views should be recomputed for every change in the definition or base relations, or they should be adapted incrementally from existing views. In this paper, we have examined several ways o performing changes in materialized views their selection and maintenance in data warehousing environments. We have also provided a comprehensive study on research works of different authors on various parameters and presented the same in a tabular manner.**

Keywords- Materialized views; view maintenance; view selection; view adaptation; view synchronization.

I. INTRODUCTION

Data warehouse is referred as a subject-oriented, non-volatile & time variant centralized repository that preserves quality data [1]. A data warehouse extracts and integrates information from diverse operational systems prevailing in an organization under a unified schema and structure in order to facilitate reporting and trend analysis. Information sources which are integrated in the data warehouse are dynamic in nature i.e. they may transform or evolve in terms of their instances and schemas. Moreover, requirements specified by the various stakeholders and developers frequently change owing to numerous reasons as mentioned below [17] [18] 19]:

1. Ambiguous or insufficient requirements during the developmental phase [17].
2. Change in the requirements during the operational phase of the Data Warehouse which results in the structural evolution of the data warehouse [18].
3. Reorganization of the data warehouse schema during the operational phase of the data warehouse as a result of different design solutions that are decided upon [18].
4. New user or business requirements arise or new versions need to be created [18] [19].
5. Periodical revisions are made in order to eliminate the errors & redundancies [17][18].
6. The data warehouse must be adapted to any changes which occur in the underlying data sources [18] [19].

Hence, data warehouse and views present in warehouse must evolve whenever there is any modification or update in the requirements or base relations, in order to fulfill the needs and constraints allocated by the various users who need the assistance of data warehouse system. In fact, data warehouse evolution process never ceases. Appropriate techniques should be devised to handle the above mentioned changes in the data sources as well as view definitions to keep the warehouse in its most consistent state.

Whenever any user poses a query, the query is processed directly at this repository thereby, eliminating the need to access the actual source of information. The resulting datasets that are generated in the response to the queries raised by the users are called as *views*, which represent functions derived from the base relations to support viewing of snapshots of stored data by the users according to their requirements. These derived functions are recomputed every time the view is called upon. Re-computing and selection of views becomes impossible for each and every query especially; when the data warehouse is very large or the view is quite complex or query execution rate is high. Thus, we accumulate some pre-calculated results (or views) in our central repository (i.e. data warehouse) in order to provide faster access to data and enhance the query performance. This technique is referred as *materialization of views.*

Materialized views act as a data cache that gather information from distributed databases and support faster and reliable availability of already computed intermediate result sets (i.e. responses to queries). Data sources in current scenario are becoming quite vast and dynamic in nature i.e. they change rapidly. Consequently, frequency of deletion, addition and update operations on the base relations rises unexpectedly. Whenever the underlying base relation is modified the

corresponding materialized view also evolves in reaction to those changes so that it can present quality data at the view level. Hence, we need certain techniques to deal with the problem of keeping a materialized view up-to date in order to propagate the changes from remote data source to the destined materialized view in the warehouse. These techniques can be broadly classified as- *view selection*, *view maintenance*, *view synchronization* and lastly, *view adaptation.* Each one of them is explained in more detail in the next section.

The layout of the paper is as follows. In section 2, we address the above mentioned techniques and also give a brief on the literatures being reviewed for the same. Section 3, presents a comparative study of the various research works explored in the previous section. Lastly, we conclude in section 4.

II. State of the Art

In this section, we describe the various techniques designed to handle the evolution of a materialized view in response to the modifications in data sources it originated from. In addition, we also discuss the literatures being reviewed in context of each and every technique.

The tasks involved in evolution of materialized views in a data warehouse can be categorized as follows:

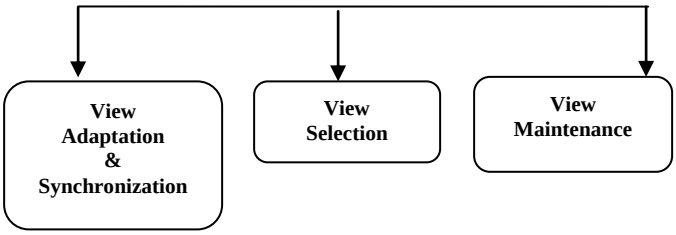

Figure 1. Tasks in materialized view evolution

A. View Adaptation & Synchronization

One of the factors that contribute to the changes in a materialized view is rewriting of views that leads to changes in the original view definition itself. This problem is addressed as *view adaptation*. Re-writing of view definitions generates the need to adapt the view schema to match it up with the most current view definition being referenced [2, 3]. View adaptation can be done either in incremental fashion or by performing full re-computation of the views [3]. If re-computation results in equivalent views then, there is no need to implement adaptation techniques because data is preserved. Non-equivalent definitions create new schema for the same view resulting in evolution of the original view. Some of the examples are listed below:

TABLE I. EXAMPLES OF SCHEMA CHANGES IN VIEW ADAPTATION

Schema changes	Description
Rename	Data preserving, no adaptation required.
Drop/Delete	Data deleted, hence non-equivalent views might be generated
Normalization	Schema structure and data preserved, hence no adaptation done.

In [2] the authors have provided a comprehensive study on various adaptation techniques. They have also provided re-definitions of all SQL clauses and views when local changes are made to view definitions. But they have only handled single materialized view changes.

In [13] author has discussed various view adaptation techniques where only the changes in view definitions cause adaptation in the views. Relation algebra binary operators can be added to SQL clauses to adapt the views. Expression trees are used to evaluate view definitions.

Another technique employed to handle materialized views is *view synchronization.* This technique changes the view definition when the structure of its base relations changes. It addresses both equivalent & non-equivalent view re-definitions [3]. Some of the changes that result in creation of new schema definitions are as follow [3]:

TABLE II. EXAMPLES OF CHANGES THAT RESULT IN SCHEMA CHANGES

Schema changes	Description
Rename	Renames the attributes and tables in the original view
Drop/Delete	Deleted attributes or tuples or tables in original views

EVE (Evolvable View Environment), a general framework has been developed in [4] to handle view synchronization in large distributed dynamic environments like- WWW. A view definition language, *E-SQL*, has also been designed along with some replacement strategies to propagate the changes in affected view components.

B. View Selection

The most important issue while designing a data warehouse is to identify and store the most appropriate set of materialized views in the warehouse so that they optimise two costs included in materialization of views: the query processing cost and materialized view maintenance cost.

Materialization of all possible views is not recommended due to memory space and time constraints [6]. The prime aim of view selection problem is to minimize either one of the constraints or a cost function as shown below:

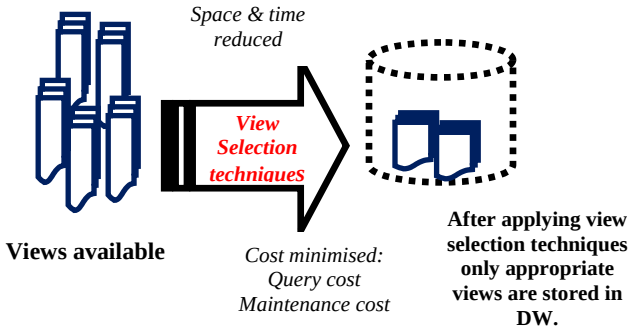

Figure 2. View Selection Process

Hence, view selection problem is formally defined as a process of identifying and selecting a group of materialized views that are most closely-associated to user-defined requirements in the form of queries in order to minimize the query response time, maintenance cost and query processing time under certain resource constraints [6].

In [5] authors have developed a AND/OR graph based approach to handle view selection problem in data cubes present in the data warehouse by taking an example of TPC-D benchmark database. They have also proposed an optimization algorithm to select certain views, but, this algorithm does not perform well in some of the cases.

Another graph based approach has been discussed in [6] in order to select a set of views for special cases under disk-space and maintenance cost constraints. AND view graphs have been discussed to evaluate the global plan for queries and OR view graphs focus on data cubes. They have proposed greedy heuristics based algorithms to handle the same. But still the approach has certain limitations like very little insight into the approximation of view-selection problem in AND/OR view graphs. Problem in AND view graphs is still not known to be NP-hard.

In [7] the authors have proposed two algorithms, one for view selection and maintenance and the second one for node selection for fast view selection in distributed environments. They have considered various parameters: query cost, maintenance cost, net benefit & storage space.

In [8] presented a framework for automatically selecting materialized views and indexes for SQL databases that has been implemented as a part of performance tuning in SQL Server 2000.

In [9] authors have presented a framework for selection of views to improve query performance under storage space constraints. It considers all the cost metrics in order to provide the optimal set of views to be stored in the warehouse. They have also proposed certain algorithms for selecting views based on their assigned weightage in the storage space and query.

In [14] a clustering based algorithm ASVMRT, based on clustering. Reduced tables are computed using clustering techniques and then materialized views are computed based on these reduced tables rather than original relations.

C. View Maintenance

Re-computation of materialized views is quite a wasteful task in data warehousing environments. Instead, we can only update a part of the views which are affected by the changes in the base relations. Hence, *View maintenance* incrementally updates a view by evaluating the changes to be incorporated in the view so that it can evolve over time. If views are maintained efficiently then, the overhead incurred while performing expensive joins and aggregations is eliminated to a larger extent.

In [10] authors have proposed a framework for dynamic environments called *DyDa*, for view maintenance in order to handle both concurrent schema and data changes. They have identified three types of anomalies and also proposed some dependency detection and correction algorithms to resolve any violation of inter-dependencies occurring between the maintenance processes.

An algorithmic approach has been implemented in [11] for incremental materialized view maintenance. The authors have employed the concept of version store so that the older versions of relations can be preserved and retrieval of correct data in the desired state is available round the clock. They have further proposed architecture to support of DW augmented with a View Manager.

In [12] authors have designed algebra based algorithm for incremental maintenance of views by schema restructuring. They have proposed a *SchemaSQL* language to handle data

updates and schema changes. Moreover, transformation operators have also been proposed to propagate data and schema changes easily.

View maintenance problem has been dealt in [3] by means of a compensation algorithm that eliminates interfering update anomalies encountered during incremental computations. Version numbers have been assigned to the updates occurring on the base relations to arrange them in a proper order. These numbers also help in detecting update notification messages that might be lost in the whole process of propagating the changes from source relation to views.

In [15] authors have presented PNUTS to handle asynchronous view maintenance in VLSD databases. The main approach is to defer expensive views by identifying RVTs & LVTs. PNUTS is also supported by a consistency model to hide details for replication of views. They have also listed the supported as well as unsupported views. Evaluation also reveals the performance of PNUTS on fault tolerance, throughput, complexity, query cost, maintenance, view staleness, latency, etc.

In [16] authors have discussed issues related to materialized views and their maintenance in Peer Data Management systems by using schema mappings (SPDMS). They have designed a hybrid peer architecture that consists of peers and super peers. Also, concepts of local, peer and global views have been developed to handle global view maintenance by handling peer vies in local PDMS, where, relations are numbered. Mapping rules guide the changes to map one version number to a new version. A push-based algorithm for view maintenance has been developed to handle view maintenance in a distributed manner.

III. COMPARATIVE STUDY

We have analyzed the various research works on several parameters and presented their comparison in the table below.

Table III. COMPARISON OF VARIOUS RESEARCH WORKS

Features / Authors	Technique	Issues addressed	Changes handled	Proposed work	Query Language Supported	Meta Data supported	Advantages	Disadvantages	Tool support / implementation
Gupta, Mumick, Rao & Ross (2001) [2]	View adaptation	Re-materialization + In-place adaptations + Non-in place adaptations	Handling of local changes in view definition	Redefinitions of all SQL clauses + Guidelines for users & DBA	SQL based view definition language	Additional information kept with materialization	Query & cost optimization	Only single materialized view changes addressed	Not addressed
Mohania (1997) [13]	View Adaptation	Materialized view adaptation in distributed DW	Changes in view definition	Expression trees + Relational binary operators + Join & derive count values	SQL clauses: Select, From, Where	Additional results also materialized	Re-computation not needed + Cost of computing decreased	Overheads in maintain additional materialized results	Not addressed
Lee, Nica & Elke (2002) [4]	View synchronization	Synchronization in distributed dynamic environments	Schema changes Of data sources	EVE framework + replacement strategies + Algorithms	E-SQL view definition language	Meta Knowledge base	Handling changes in large dynamic environment (WWW) + A general framework	Only addressed schema changes in sources + No cost and quality issues addressed	JAVA + JDBC + MS-Access
Dhote & Ali (2007) [5]	View Selection	Selection of views to minimize query response time	Data cube changes	AND/OR DAG to minimize the query response time + Optimization algorithm	SQL based	✘	Heuristic based algorithm + Simple approach	Algorithm does not works well on certain cases + Cant be used on whole data cube (works only on lattice)	Not addressed
Gupta & Mumick (2005) [6]	View Selection	View selection under disk space & maintenance cost constraints.	Global evaluation plan for queries + Data cubes	AND/OR view graphs + Greedy heuristics based algorithms	SQL based	✘	Optimal solution for special cases (AND/OR views) + Polynomial time heuristics	Approximation in view-selection problem not addressed + Problem in AND view graphs not NP-hard + Solution fairly close to optimum	Not mentioned
Karde & Thakare (2010)	View Selection	Query cost, maintenance cost, storage space &	In distributed environments	Algorithm for creation and maintenance of views	Not mentioned	✘	Query performance improved	Only distributed environments highlighted	Not addressed

[7]				+ Algorithm for node selection					
Agrawal, Chaudhari & Narasayya (2000) [8]	View Selection	Automated view and index selection	✘	Framework for index & view selection + Candidate selection & enumeration techniques	SQL based	✘	Robust tool support + Both indexes & view selected	Only a part of physical design space addressed	SQL Server 2000
Ashadevi & Balasubramanian [9]	View selection	Cost-effective view selection under storage space constraints	✘	Framework for selecting views + Algorithm for the same + Cost metrics	Not addressed	✘	All cost metrics considered	Query response time not considered + Threshold value not indicated clearly	Algorithms implemented in JAVA
Yang & Chung (2006) [14]	View selection	Attribute-value density + Clustered tables + Selection of views based on clustered /reduced tables	Related dimensions or relations	ASVMRT algorithm for view selction	SQL based	✘	Faster computation time + Reduced storage space + 1.8 times performance better than conventional algorithms	Maintenance of reduced tables not addressed + Updating Reduced tables needs attention	In pubs database + ETRI
Chen, Zhang & Elke (2006) [3]	View maintenance	Source Data updates + Preserving & non-preserving schema changes + 3 types of anomalies	Source schema & data updates	DyDa Framework + Dependency & Correction algorithms	SQL based maintenance & compensation queries	✘	Can handle concurrent & interleaved data and schema changes	Extra cost on data updates + Cannot maintain mixed updates in single process	JAVA & Oracle 8i
Almazyad & Siddiqui (2010) [10]	View maintenance	Incremental view maintenance + synchronization between DW and source + lost update notifications	Source relation changes	Framework with version store	Not mentioned clearly	Version store provides needed metadata	Synchronization between source and DW + Detection of update notification messages	Process becomes a bit lengthy + more space needed + Version numbers should be handled properly	Not addressed
Koeller & Rundensteiner (2004) [11]	View maintenance	Schema restructuring of views	Data + Schema changes	Algebra based maintenance + transformation operators	Schema SQL queries	Not clearly mentioned	Algebra-based can be adapted to other query languages easily	Time consuming process	JAVA + Oracle 8 (JDBC)
Ling & Sze (2001) [12]	View maintenance	Update anomalies + Notification messages	Modifications in base relations	Compensating algorithms + Version numbers	Not addressed	Present in form of log files + version numbers	Algorithms does not require quiescent state before views can be refreshed	Time consuming + Version numbers should be handled	Not addressed

							+ Update notifications handled efficiently + version numbers reflect state of relations	properly	
Agrawal, Silberstein, Cooper, Srivastava & Ramakrishnan (2009) [15]	View maintenance	Asynchronous view (Remote view Tables RVT, Local View Tables LVT) + Replication of views	Deferred Indexes & Views in Very Large Scale Distributed databases, horizontally partitioned	PNUTS + consistency model + RVTs &LVTs + Group-by views, select views, indexes & equi-join views	SQL based	Metadata for view definitions & partitions	Improved client latency + Improved scalability + Balanced view staleness, system complexity + Improved query cost	Views maintenance adds load to system + VLSD are complex + Decreased throughput + Complex failure recovery	C++ + FreeBSD 6.3 (Linux can also be used)
Qin, Wang & Du (2005) [16]	View Maintenance	Global view maintenance by maintaining local views in PDMS	Schema changes + Schema mappings amongst peers	Hybrid peer architecture (P2P & Peer-super peer) + Local, global & peer views + Rules to use updategrams & boosters + Push-based Algorithm	Not mentioned	One kind of Super peer maintains metadata for mapping schemas in intre-peers or inter-peer changes in local PDMS	Decentralised maintenance strategy + Higher efficiency + Parallelism + Efficient in 80-20 distribution + Central bottlenecks avoided	Information sharing is complex & difficult in PDMS + Querying not addressed	Simulation system developed in JAVA

IV. CONCLUSION

In this paper we have presented an analysis of different approaches being proposed by various researchers to deal with the materialized views in data warehouse namely- view adaptation & synchronization, view selection and view maintenance. We have examined these techniques on various parameters and provided a comparative study in a tabular manner.

V. FUTURE WORK

As future work, we will direct our research towards batch-oriented view maintenance and selection strategies. A thorough investigation of the methodologies to handle materialized views in highly distributed environments for query processing and analysis seems worth attention.

REFERENCES

[1] W. Inmon, "Building the data warehouse", Wiley publications, pp 23, 1991.

[2] A. Gupta, I. Mumick, J. Run, and K. Ross, "Adapting materialized views after redefinitions: techniques and a performance study", In Elsevier Science Ltd., pp 323-362, 2001.

[3] S. Chen, X. Zhang, and E. Rundensteiner, "A compensation based approach for view maintenance in distributed environments", In IEEE transactions and data engineering, 18, 2006.

[4] A. Lee, A. Nica, and E. Rundensteiner, "The EVE approach view synchronization in dynamic distributed environments", In IEEE Transactions and Data Engineering, 14, 2002.

[5] C. Dhote, and M. Ali, "Materialized view selection in data warehouse", In International Conference on Information Technology, 2007.

[6] A. Gupta, and I. Mumick, "Selection of views to materialize in a data warehouse", In IEEE Transactions on Knowledge and Data Engineering, vol. 17, 2005.

[7] P. Karde, and V. Thakare," Selection of materialized views using query optimization in database management: An efficient methodology", In International Journal of Management Systems, vol. 2.

[8] S. Agrawal, S. Chaudhari, and V. Narasayya, "Automated selection of materialized views and indexes for SQL databases", In Proceedings of 26th International Conference on Very Large Databases, 2000.

[9] B. Ashadevi, and R. Balasubramanian, "Cost effective approach for materialized views selection in data warehouse environment", In International Journal of Computer Science and Network Security, vol. 8, 2008.

[10] A. Almazyad, and M. Siddiqui, "Incremental view maintenance: an algorithmic approach", In International Journal of Electrical & Computer Sciences, vol. 10, 2010.

[11] A. Koeller, and A. Rundensteiner, "Incremental maintenance of schema–restructuring view in SchemaSQL", In IEEE Transactions and Data Engineering, 16, 2004.

[12] T. Ling, and E. Sze," Materialized view maintenance using version numbers", In Proceeding of Springer Berlin, 2001.

[13] M. Mohania, "Avoiding re-computation: View adaptation in data warehouses", in australian research council, 1997.

[14] J. Yang, and I. Chung, "ASVMRT: Materialized view selection algorithm in data warehouse", In International Journal of Information Processing System, 2006.

[15] P. Agrawal, A. Silberstein, B. Cooper, U. Srivastava, and R. Ramakrishnan, "Asynchronous view maintenance in vlsd databases", In SIGMOD International Conference on Management of Data, 2009.

[16] B. Qin, S. Wang, and X. Du, "Effective maintenance of materialized views in peer data management systems", In Proceedings of First International Conference on Semantics, Knowledge and Grid, 2005.

[17] D. Sahpaski, G. VelInov, B. Jakimovski, and M. Kon-Popovska, "Dynamic evolution and improvement of data warehouse design", In Balkan Conference in Informatics, 2009.

[18] B. Bebel, J. Eder, C. Koncilia, T. Morzy, and R. Wrembel, "Creation and management of versions in multiversion data warehouse", In Proc. ACM SAC, 717–723, 2004.

[19] B. Bebel, J. Eder, C. Koncilia, T. Morzy, and R. Wrembel, "Formal approach to modeling data warehouse", In bulletin of the Polish Academy of Sciences, 54, 1, 2006.

[20] Vashishta, S. (2011). Efficient Retrieval of Text for Biomedical Domain using Data Mining Algorithm. International Journal of Advanced Computer Science and Applications - IJACSA, 2(4), 77-80.

[21] Hanandi, M., & Grimaldi, M. (2010). Organizational and collaborative knowledge management : a Virtual HRD model based on Web2 . 0. International Journal of Advanced Computer Science and Applications - IJACSA, 1(4), 11-19.

AUTHORS PROFILE

Garima Thakur is pursuing her M.Tech in Information Technology from Guru Gobind Singh Indraprastha University, Delhi, India. She has done her B.Tech in Computer Science branch from the same university in the year 2009. She is doing her research work in the field of Data warehouse & Data Mining and Knowledge Discovery.

Dr. (Mrs.) Anjana Gosain is working as reader in University school of information technology. She obtained her Ph.D. from GGS Indraprastha University & M.Tech in Information Systems from Netaji Subhas Institute of Technology (NSIT) Delhi. Prior to joining the school, she has worked with computer science department of Y.M.C.A institute of Engineering, Faridabad (1994-2002). She has also worked with REC kurukshetra. Her technical and research interests include data warehouse, requirements engineering, databases, software engineering, object orientation and conceptual modeling. She has published 18 research papers in International / National journals and conferences.