Issues of Materialized Views in Datawarehousing for Efficient Management of Information

G.Prabagaran
Registrar,
Dept of Students' Affairs,
Salalah College of Technology,
Salalah Sultanate of Oman

N. Venkatesan
Associate Professor,
Dept of IT,
Bharathiyar College of Engg. & Technology,
Karaikal

ABSTRACT
Multiple Views can be formed and materialized from a data warehouses per the user requirements specified in the queries being generated against the information contained in the warehouse. Selecting Views to be materialized is one of the most important decisions in designing a warehouse. Due to change in user requirements and constraints over time View definitions stored in a data warehouse are dynamic in nature. This paper focus on the issues of materialized views in Data warehousing to enable efficient Information Management. The issues include selection, maintenance and updating of Materialized views and how these issues create an impact in Business scenarios.

Keywords: Data warehouse, Views, Information management

1. INTRODUCTION
A datawarehouse 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. This paper is organized in four sections. The first section discuss the view selection issue of the Materialized views. The view Selection Issue deals with the problem to select a set of derived views to materialize that minimizes the sum of total query response time & maintenance of the selected views. The Materialized View is like a cache i.e. a copy of data that can be accessed which is required for frequent set of queries. Section two discuss about the issues in maintaining the Materialized views.

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. Section three discuss about the issues in updating the materialized views. The updates to the base relation which are not affecting the views are filtered and treated as irrelevant updates. A differential algorithm can be applied to reevaluate the view expression for the remaining database updates..

This paper is organized as follows: Section 2 describes about the view selection for materialization. Materialized view maintenance process is discussed in Section 3. Delta table is described in section 4. Updating issues and lazy view maintenance is organized in the chapter 5. Section 6 concludes along with future work.

2. VIEW SELECTION FOR MATERIALIZATION
Selecting Views to be materialized plays a vital role in data warehousing environment for decision making. Different factors are taken into consideration for efficient view selection. Quick response time and reliable results for frequent queries are most important factors in selecting the views to be materialized. The activities for materialized view management are: 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 [1,2]. 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 [2,5,6]. Unlike answering queries using views that need to handle ad-hoc 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.

Two algorithms are discussed 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. 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[3].

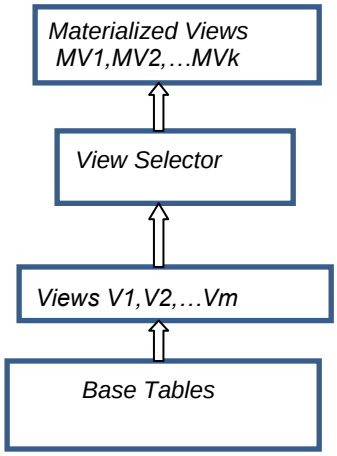

Figure1: Materialized View Selection Process

3. MATERIALIZED VIEW MAINTENANCE

Materialized views improve the performance of query processing in terms of speed and response time. This can be achieved to larger extent by maintaining the materialized views. Two basic categories maintenance are analyzed one is eager maintenance and other is lazy maintenance. In the case of eager view maintenance updates in the base tables reflects eagerly in the views which leads to poor response time for updates. To address this situation some database systems also support deferred maintenance where maintenance of a view is delayed and takes place only when explicitly triggered by a user. This approach has the serious drawback that a query may see an out-of-date view and produce an incorrect result. A solution to the above mentioned eager and deferred view maintenance schemes are lazy view maintenance.

In lazy maintenance updates do not maintain views but just store away enough information so that affected views can be maintained later. Actual maintenance is done by low-priority jobs running when the system has free cycles available. If the system has enough free cycles and a view is maintained before it is needed by queries, neither updates nor queries pay for view maintenance. If a view is not up to date when needed by a query, it is transparently brought up to date before the query is allowed to access it. One of the advantage of lazy maintenance is it allows updates to complete faster so locks are released sooner. It reduces the frequency of lock contention, lock conflicts and transaction aborts. The updates that affect highly aggregated views are benefited by the lazy view maintenance because they tend to have higher rates of lock conflicts.

3.1. Lazy View Maintenance System Design

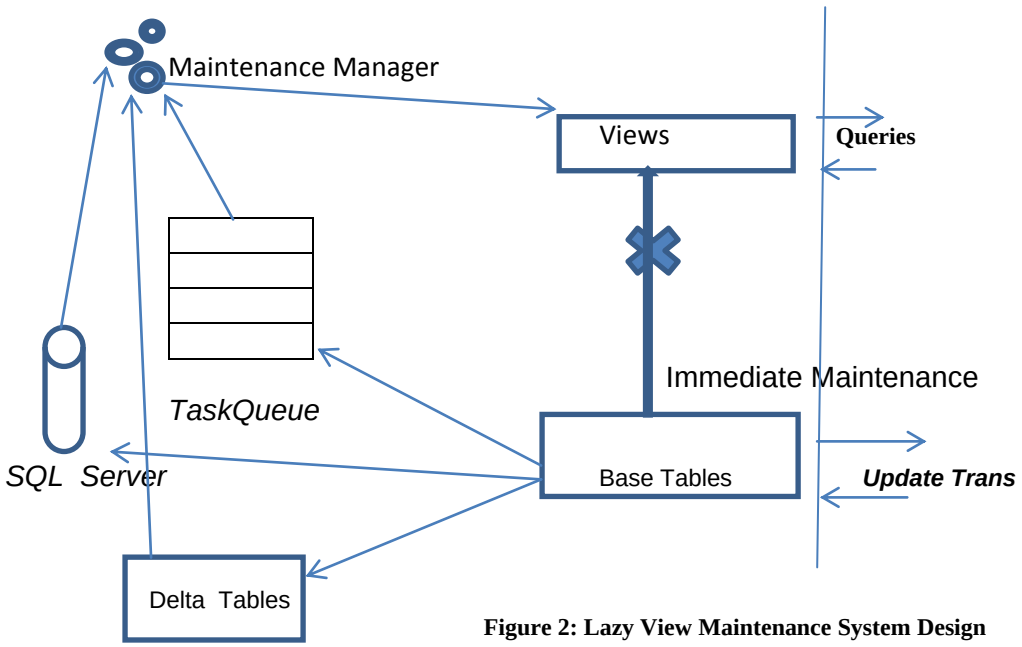

Figure 2: Lazy View Maintenance System Design

4. USAGE OF DELTA TABLES

Execution of SQL data manipulation statement like insert, delete, or update against a base table produces a stream of delta rows. The delta stream is then transformed into a split delta stream with an additional action column Action. Each delta row in the split delta stream encodes what change was made to a row of the target base table. The action column indicates if the delta row represents an insert, delete, or update of a row. In a split delta stream , an update is represented by two delta rows, one containing the old values with action "delete", the other containing the new values with action "insert". For each base table we create a corresponding delta table which stores the split delta stream for its base table. Rows in the delta table include two additional columns, the transaction sequence number TXSN and the statement sequence number STMTSN that indicate which transaction and statement produced the delta row. A delta table is clustered on columns TXSN, STMTSN, Action, plus the primary key columns of its base table. A maintenance task specifies which view needs to be maintained, the set of updated base tables, the transaction sequence number (TXSN) and the commit sequence number (CSN) of the originating transaction. Task status is used by the maintenance manager to schedule and track individual tasks.

4.1. MAINTENANCE MANAGER

This component keeps track of active view maintenance tasks and what database versions and delta streams are needed. It is also responsible for constructing view maintenance jobs and scheduling them. To be able to quickly find all maintenance tasks for a given view, the manager maintains a hash table containing an entry for each materialized view with active maintenance tasks. Each entry has a linked list containing the maintenance tasks of the view. The list is sorted in an increasing order on commit sequence number.

4.2. TASK TABLE

Maintenance tasks are also stored persistently in a global view-maintenance task table. The table is used for recovery purposes only, not for normal processing. A maintenance task is added to this table as part of the transaction that generated it and deleted as part of the transaction that performs the maintenance.

5. UPDATION ISSUES DURING EAGER AND LAZY VIEW MAINTENANCE

During update statement to a base table referenced by a number of materialized views Eager maintenance updates all materialized views that reference the base table immediately after the update statement. In the case of Lazy maintenance view maintenance is skipped. Instead, enough information is saved so the affected views can be updated later. The split Delta stream produced by the update statement is appended to the corresponding delta table.

Versioning is enabled so that when the update is applied to the base table, the old version of each modified row is stored in the version store. Multiple update statements together form an update transaction. The transaction internally maintains a log which keeps the status of table modification and the statement responsible for the modification. Each update statement reports its own information at the end of its execution. When the update transaction commits, maintenance tasks are constructed based on the information reported during execution. One maintenance task is generated per affected materialized view. The tasks are then passed on to the maintenance manager and also written to the persistent task table. If the update transaction aborts, no information is saved and no maintenance tasks are constructed.

5.1. DECIDING FACTOR FOR EAGER AND LAZY VIEW MAINTENANCE

Application is the major deciding factor for selecting eager or lazy view maintenance for a particular view. The other factors for the choice of maintenance strategy for a materialized view is as follows

1. The ratio of updates to queries and how soon queries follow after updates.
2. The number of rows affected by each update.
3. View maintenance cost.

Eager Maintenance is suitable for materialized views whose base tables are seldom updated and the updates are likely to be followed immediately by queries. It is also suitable for views. Where the maintenance cost is relatively low. Lazy maintenance is suitable for views with more frequent small updates and whose maintenance costs are relatively high.

6. CONCLUSION

This paper discussed about the need of materialized Views in Data warehousing environment. Various issues of materialized view in data warehousing environment is discussed. Selection of Materialized view is discussed. Materialized view maintenance issue is discussed by addressing the types of View maintenance. Various factors that supports for deciding optimal view maintenance schemes are discussed. The impact of update statement in View Maintenance is also addressed. This paper discussed on the issues of Materialized views in data warehousing environment for the Information management Community.

REFERENCE:

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

[2] 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

[3] Mr.P.PKarde and Dr.V.M. Thakare "Selection of Materialized View Using Query Optimization In Database Management: An Efficient methodology International Journal of Database Management Sysems(IJDMS),Vol.2, No.4, November 2010 DOI: 10.5121/ijdms.2010.2410 116.

[4] J.Goldstein and P.A.Larson.Optimizing queries using materialized views :A practical, scalable Solution. In Proceedings of SIGMOD Conference, 2001.

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

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

[7] T.Griffin and L.Libkin.Incremental maintenance of views with duplicates. In Proceedings of

SIGMOD Conference, 1995.

[8] A.Gupta.I.S. Mumick.and V.S. Subrahmanian. Maintaining views incrementally. In Proceedings Of SIGMOD Conference, 1993

[9] P.A.Larson and J.Zhou. Efficient maintenance of materialized outer-join views. In Proceedings of ICDE Conference, 2007.

[10] W. Lehner. R.Sidle.H. Pirahesh and R.Cochrane. Maintenance of automatic summary tables in Proceedings of SIGMOD Conference, 2000.