Data Extraction Using Materialized Views

Yupar Kyaw Lwin, Aye Thida
University Of Computer Studies, Mandalay
yuyathu@gmail.com

Abstract

A data warehouse uses multiple materialized views to efficiently process a given set of queries. Materialized views selection is one of the crucial decisions in designing a data warehouse for optimal efficiency. The idea of reusing materialized results of previous data mining queries may lead to improve the performance of the system. In creating the materialized views, SQL statements are automatically generated from the system to create view table and to entry data. However, Query views need to select the desired field names and table names from the database. Materialized views are required to achieve faster query in the data warehouse. This paper also compares the execution time of the materialized view and database query view. To get fast answer, creating materialized views on photo studio data warehouse is the best.

1. Introduction

Data mining is the process of extracting or mining knowledge, requires the computation of many aggregate functions over large amounts of data. Multidimensional data analysis requires the computation of many aggregate functions over large amounts of data [1]. A data warehouse contains multiple views where a view is a derived relation defined in terms of base (stored) relations. The views stored in the data warehouse are referred to as the materialized views. Materialized views are physical structures that improve data access time by precomputing intermediary results.

In a dynamic environment choosing suitable set of views to materialize is not such an easy task, this includes more factors to be taken into consideration. Data mining aims at discovery of useful patterns from large databases or warehouses.

The results of the previous queries along with the queries are stored as a materialized view which is reused to answer the data mining queries, theses views need to be maintained with the corresponding objects. Refreshing is an important concept for maintaining accuracy, it deals with the refreshing or sync of views to their objects. The ability to sync with the object increases the accuracy but it may cause delays to answer queries, so a trade off must be established for optimal results [8].

2. Related Work

The problem of finding views to materialize to answer queries has traditionally been studied under the name of view selection. Its original motivation comes up in the context of data warehousing. A greedy algorithm is presented for the selection of materialized views so that query evaluation costs can be optimized in the special case of "data cubes" [2].

However, the costs for view maintenance and storage were not addressed in this piece of work. A heuristic algorithm which utilizes a Multiple View Processing Plan (MVPP) is presented to obtain an optimal materialized view selection, such that the best combination of good performance and low maintenance cost can be achieved [2]. However, this algorithm did not consider the system storage constraints.

A greedy algorithm is developed to incorporate the maintenance cost and storage constraint in the selection of data warehouse materialized views [2]. "AND-OR" view graphs were introduced to represent all the possible ways to generate warehouse views such that the best query path can be utilized to optimize query response time.

The use of an evolutionary algorithm is explored for materialized view selection. A modified genetic algorithm is proposed for the selection of a set of views for materialization.

3. Materialization Framework

For any kind of performance improvement system based on locally materializing or caching data, a framework is required for representing and using the materialized data. A framework is presented for selecting views to materialize so as to achieve the best combination of good query response, low query

processing cost and low view maintenance cost. For database system caches, tuple based schemes or recent approaches are based on semantic caching [6].

Data warehouse is capable of answering queries and performing analysis in an efficient and quick manner, in the view of the fact that integrated information is directly available at the warehouse with differences already resolved [2].

The most important tribulation in data warehousing is the materialization of views. Materializing all possible views is impractical, which entails large computation and space. Therefore, the key issue in data warehousing is to select appropriate set of views to materialize that hit a steadiness between computational cost and increased query performance, commonly referred as "view selection problem"[2]. Cluster queries to reduce the search space for the materialized view selection problem [7].

3.1. Selecting Data to Materialize

Several factors affect the choice of materialized views. Therefore selecting the suitable views to materialize has become an imperative issue in warehouse implementation. The view selection problem is to select a set of views to be materialized, that minimize total cost associated with materialized views under storage space and maintenance cost constraints.

It is the question of identifying what is the portion of data that is most useful to materialize. There are several factors that can be analyzed or must be considered for identifying such data. The classes of data that are most frequently queried by users are obviously good candidates for materializing locally.

3.2. Views and Data Warehouses

A data warehouse holds multiple views and we have referred the materialized views as the views stored in the data warehouse. Materialized views are physical structures that precompute the intermediary results, thereby improving data access time. However, additional storage space and maintenance overhead when refreshing the data warehouse is necessitated by the employment of materialized view. Owing to the direct availability of integrated information at the warehouse with differences already resolved, data warehouse has the ability to answer queries and perform analysis efficiently and quickly [3].

The Data Warehouse itself is considered to be a materialized view of the operational databases and external data sources. Fast execution of decision support queries is frequently accomplished by materializing views on data warehouse base tables. While deciding which views to materialize, one should consider the following issues:

- how many queries potentially can be speed up,
- how much space will be required to store the views and
- how will the views influence the data warehouse maintenance (update).

3.3. Materialized Views

A view is a virtual table that consists of data that is pulled out of one or more existing tables by a query [4]. If stored as a file, a view is called a materialized view. Materialized data mining views are physical data warehouse structures, created explicitly or implicitly, used to store precomputed results of selected data mining queries.

Additional advantage of materialized views is the fact that data mining usually takes place in a data warehouse where changes to base relations (and thus to the stored patterns) do not happen continually over time but are accumulated and loaded to the data warehouse during data warehouse refresh process.

The patterns discovered and stored in the materialized view remain valid for a long period of time until next data warehouse refresh. Validation of patterns can be postponed until next warehouse refresh event. In a relational database management system, a view is a virtual table representing the result of a database query. A materialized view is the view cached in a real table [5]. Selecting views to approach, the query result is stored as a concrete table that may be updated from the original base tables from time to time. Three types of materialized views are (1) read only, (2) updateable, and (3) writable.

These views are accessed by read-only queries and need to be maintained after updates to base tables.

4. System Overview

In this system, there are five dimension tables and one fact table. The dimension tables are item table, category table, city table, township table and customer table. The sales table is a fact table.

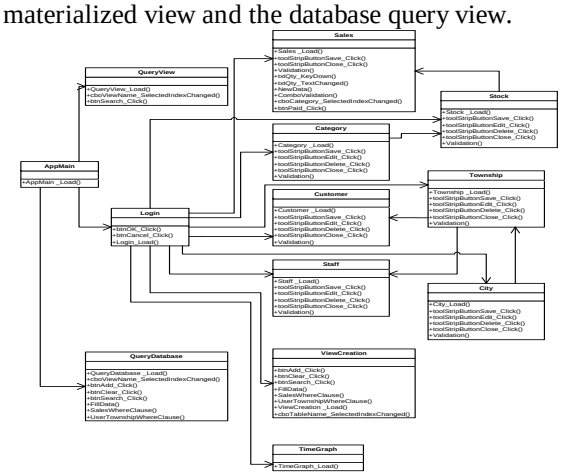
In creating the materialized view, the view name is initially given without iteration. Then the required table names are chosen and the required field names are selected and added. Finally, all the information for the materialized view is saved to review at the later time with this view name.

To review the previous query view, the materialized views are required. Even if the view name is given, the SQL statements are automatically generated from the system by using the materialized view. After the materialization, the work-flow of the DBMS continues as usual and, as a result, the query is executed as if all patterns and models are stored in the database.Unlike the materialized views, query views need to select the desired field names and choose the table names from the original database whenever the previous query views are required.

So, if the faster query view is required, the materialized view must be used. According to this point, the query analysis time is compared in this paper. However, both views can generate the desired result for only one field such as "InvoiceNo=20".

5. Implementation Results

The system implements many classes for the materialized view and the database query view.

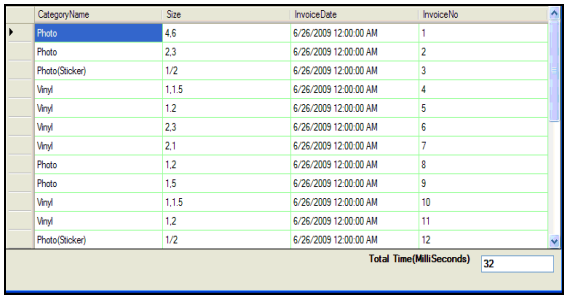


"Figure 1. Class diagram of the system"

Figure 1 shows the class diagram of the system. The main class derives three classes which are Login, QueryDatabase and QueryView.

The Login class derives nine classes. They are Township class, City class, Customer class, Staff class, Category class, Stock class, Sales class, ViewCreation class and TimeGraph class. These classes can be performed by the administrator or the staff to entry the data.

The class QueryDatabase is the class for direct query view which includes load method, table name and field names select method, add method, clear method, search method and query process method. The QueryView class is the class for the materialized views. It consists of three functions which are load function, view name select function and search function.



CategoryName	Size	InvoiceDate	InvoiceNo
Photo	4,6	6/26/2009 12:00:00 AM	1
Photo	2,3	6/26/2009 12:00:00 AM	2
Photo(Sticker)	1/2	6/26/2009 12:00:00 AM	3
Vinyl	1,1.5	6/26/2009 12:00:00 AM	4
Vinyl	1.2	6/26/2009 12:00:00 AM	5
Vinyl	2,3	6/26/2009 12:00:00 AM	6
Vinyl	2.1	6/26/2009 12:00:00 AM	7
Photo	1,2	6/26/2009 12:00:00 AM	8
Photo	1,5	6/26/2009 12:00:00 AM	9
Vinyl	1,1.5	6/26/2009 12:00:00 AM	10
Vinyl	1,2	6/26/2009 12:00:00 AM	11
Photo(Sticker)	1/2	6/26/2009 12:00:00 AM	12

Total Time(MilliSeconds) 32

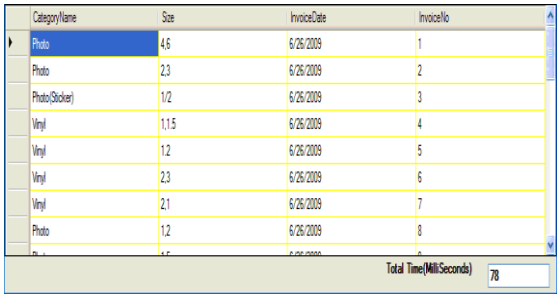
"Figure 2. View Results from materialized view for C1"

In Figure 2, the user input Select * From C1 to view the query result where C1 is the previous query view name. C1 includes the fields of CategoryName from Category table, Size from Item table, InvoiceDate from InvoiceData and InvoiceNo from InvoiceData. The system can execute the selected view name form the materialized views and the query results are displayed to the users within 32 milliseconds.

The execution time for materialized view is the time take from searching the query to retrieving the required result from the database.

```
SELECT
Category.CategoryName,Item.Size,InvoiceData.InvoiceDate,
InvoiceData.InvoiceNo FROM
Category,Item,InvoiceData,InvoiceDetail WHERE
Item.CategoryID=Category.CategoryID AND
InvoiceData.InvoiceNo=InvoiceDetail.InvoiceDetail
InvoiceNo AND Item.ItemID=InvoiceDetail ItemID
```

"Figure 3. Query statements for query view for C1"

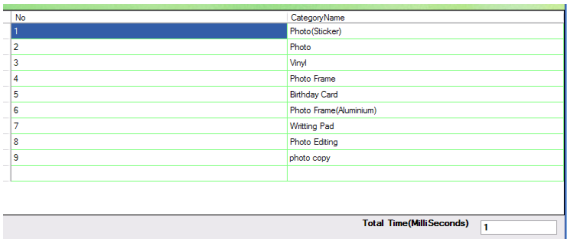


CategoryName	Size	InvoiceDate	InvoiceNo
Photo	4,6	6/26/2009	1
Photo	2,3	6/26/2009	2
Photo(Sticker)	1/2	6/26/2009	3
Vinyl	1,1.5	6/26/2009	4
Vinyl	1,2	6/26/2009	5
Vinyl	2,3	6/26/2009	6
Vinyl	2,1	6/26/2009	7
Photo	1,2	6/26/2009	8

Total Time(MilliSeconds) 78

"Figure 4. View Results from Database for C1"

After selecting the field names and table names, the SQL statement in Figure 3 is automatically generated from the system. In Figure 4, the query results from the database are displayed to the users within 78 milliseconds.



No	CategoryName
1	Photo(Sticker)
2	Photo
3	Vinyl
4	Photo Frame
5	Birthday Card
6	Photo Frame(Aluminium)
7	Writting Pad
8	Photo Editing
9	photo copy

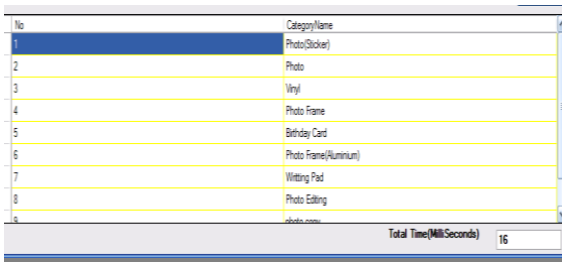
Total Time(MilliSeconds) 1

"Figure 5. View Results from materialized view for Category View"

Similarly, reviewing the query results from Category View can be found as shown in the Figure 5, 6 and 7.

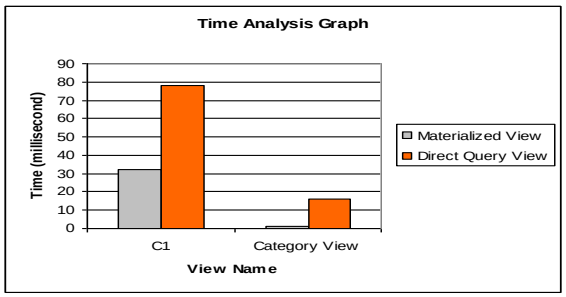
```
Select * from Category View
```

"Figure 6. Query statements for query view for Category View"



"Figure 7. View Results from Database for Category View"

The execution time for direct query view is the time taken from searching the query of user-selected tables and fields from auto-generating the SQL statements from the database. When the execution time is discussed, Core Duo 1.60 GHz processors, 1GB RAM and SQL database are used.



"Figure 8. View Results from materialized view"

The execution time for the direct query view and the materialization view of C1 and Category View is compared in Figure 8. In the analysis graph, X-axis describes view name, C1 and Category View. Y-axis describes the execution time in milliseconds.

As shown in Figure 8, the processing time of the materialized views is faster than that of the direct query view because the former requires only choosing the view name to review the previous query results and the latter must perform to choose the table names and to select the field names from the original database during the query processing.

6. Conclusion

The selection of views to materialize is one of the most important issues in designing a data warehouse. The materialized view can generate the query results for the previous query views by only choosing the view name. The query processing time for the materialized view is fast because there is no need to choose table names and select field names.

The traditional query processing cannot perform like the materialized view in reviewing the previous query view. A framework for selecting views has been presented to materialize so as to achieve the best combination of good query response, low query processing cost and low view maintenance cost in a given storage space constraints.

The most cost effective views have been selected for materialization by the framework and the maintenance, storage and query processing cost of the views have been optimized. The materialized view can satisfy every data extraction.

7. References

[1] Amit Shukla, Prasad M. Deshpande, Jeffrey F.Naughyon, "Materialized view selection for multidimensional datasets," in Proc. 24th Int. Conf. Very Large Data Bases, 1998, pp. 488–499.

[2] B.Ashadevi and Dr.R.Balasubramanian, "Cost Effective Approach for Materialized Views Selection in Data Warehousing Environment," IJSCNS International Journal of Computer Science and Network Security, VOL.8 No.10, October 2008.

[3] B.Ashadevi, "Optimized Cost Effective Approach for Selection of Materialized Views in Data Warehousing," JCS&T Vol. 9 No. 1, April 2009.

[4] Fujian Liu, "Database Server Workload Characterization in an E-commerce Environment", A Thesis Submitted to the College of Graduate Studies and Research in Partial Fulfillment of the Requirements For the Degree of Master Science in the Department of Computer Science University of Saskatchewan, December 2005.

[5] Joanna Jozefowska, Agnieszka Lawrynowicz, and Tomasz Lukaszewski, "Materialized views in mining ontology instances," Institute of Computing Science Poznan University of Technology.

[6] Naveen Ashish, "Optimizing Information Mediators By Selectively Materializing Data," A Dissertation Presented to the Faculty of The Graduate School, University of Southern Calfornia, March 2000.

[7] Nodira Khoussainova, Magdalena Balazinska, Wolfgang Gatterbauer, YongChul Kwon and Dan Suciu, "A Case for A Collaborative Query Management System".

[8] Vahida Attar and Vandana Inamdar, "Materialized Views in Data Mining," IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.12, December 2007.