Creation of Query View Using Materialized View

Khin Mar Wai
Computer University (Monywa) Myanmar
pollya.nar09@gmail.com

Abstract

A data warehouse is a redundant collection of data replicated from several possibly distributed and loosely coupled source databases organized to answer On-Line Analytical Processing (OLAP) queries. OLAP queries are typically aggregate queries over very large fact tables of Data Warehouse. This paper can create tables by organizing data that are mostly used by users and. can also show time comparison between the time taken to retrieve data from the original database and the time taken to retrieve data from the materialized views. To get fast answers, creating materialized views on that data warehouse is the best. A materialized view is the precalculated (materialized) result of a query. Materialized Views are used when immediate response is needed.

1. Introduction

This paper implemented materialized view creation, i.e. a table will be created to store the tuples resulting from the view definition and while executing the queries on the view, the tuples will be taken from the table instead of generating those on the fly as in case of normal views. This thesis had also added the materialized view maintenance functionality, wherein, as and when updates are made to the base tables, the materialized view will also be updated.

The advantage of using materialized views is that for the retrieval type of queries which may have to be performed quite frequently, the base table need not be queried each time. Instead, a simple query on the view would retrieve all the required data include the aggregate functions like max, min, avg, sum etc. In case of the standard views, which are currently available in PostGres, whenever data is retrieved from the view, the query is actually fired on the base table on which the view is created. Hence, this is a costly operation. This can be avoided using materialized views which are extremely beneficial for aggregate type of queries.

Data warehousing and On-line Analytical Processing (OLAP) are essential elements of decision support, which has increasingly become a focus of the database industry. Many commercial products and services are now available, and all of the principal database management system vendors now have offerings in these areas. Decision support places some rather different requirements on database technology compared to traditional on-line transaction processing applications. This paper provides an overview of data warehousing and OLAP technologies, with an emphasis on their new requirements. This thesis describes back end tools for extracting, cleaning and loading data into a data warehouse, multidimensional data models typical of OLAP, front end lient tools for querying and data analysis, server extensions for efficient query processing and tools for metadata management and for managing the warehouse. This system provides mechanisms of implementing materialized views to retrieve and get fast answer for a given query. The most common situations where you would find materialized views useful are in data warehousing applications and distributed systems. In warehousing applications, large amounts of data are processed and similar queries are frequently repeated. If these queries are pre-computed and the results stored in the data warehouse as a materialized view, using materialized views significantly improves performance by providing fast lookups into the set of results.

2. Related Word

According to W.H.Inmon, a leading architect in the construction of data warehouse systems, a data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process.

G.Limaye presents that an enterprise data warehouse contains historical detailed data about the organization. Typically, data flows from one or more online transaction processing (OLTP) databases into the data warehouse on a monthly, weekly, or daily basis. The data is usually processed in a staging file before being added to the data warehouse. Data warehouses typically range in size from tens of gigabytes to a few terabytes, usually with the vast majority of the data stored in a few very large fact tables.

S.Chaundhuri and U.Dayal discussed the techniques employed in data warehouses to improve performance is the creation of summaries, or aggregates. They are a special kind of aggregate view which improves query execution times by precalculating expensive joins and aggregation operations prior to execution, and storing the results in a table in the database. For example, a table may be created which would contain the sum of sales by region and by product.

3. Theory Background

Data warehouses have been defined in many ways, making it difficult to formulae a rigorous definition. Loosely speaking, a data warehouse refers to a database that is maintained separately from an organization's of a variety of application systems. Data warehouse is a relational database that is designed for and analysis rather that for transaction processing. They support information processing by providing a solid platform of consolidated historical data for analysis.

A data warehouse can be view as an information system with the following attribute:

1. It is a database design for analytical tasks using data multiple applications.
2. It supports a relatively number of users with relatively long interactions.
3. Its usage is read intensive.
4. Its content is periodically updated.
5. It contains current and historical data to provide an analytical perspective of information.
6. It contains large few tables.
7. Each query frequently results in a large set and involves frequent full scan and multiple joins.

A data warehouse system comprises the data warehouse and all components used for building and accessing the data warehouse. Data warehouse is a repository of an organization's electronically stored data. This definition of the data warehouse focuses on data storage. However, the means to retrieve and analyze data, to extract, transform and local data, and to manage the data dictionary are also considered essential components of a data warehousing system.

Data warehouse systems serve users or knowledge workers in the role of data analysis and decision making. Such systems can organize and present data in various formats in order to accommodate the diverse need of the different users. These systems are known as On-Line Analytical Processing (OLAP) systems. Data warehouses are optimized for speed of data retrieval. Frequently data in data warehouses are demoralized via a dimensional-based model. Also, to speed data retrieval, data warehouse data are often stored multiple times – in their most granular form and in summarized forms called aggregates. Data warehouse data are gathered from the operational systems and held in the data warehouse even after the data has been purged from the operational systems.

ROLAP and MOLAP are simply terms that refer to common methods for storing data used by OLAP systems. In either case, the user interface is still an OLAP interface. The only difference is the database technology used to store their data in a series of tables and columns. Multidimensional database store their data in large multidimensional arrays. The most common data structure for ROLAP data is the star schema or some variant, like the snowflake schema. ROLAP in which relational database system tuned for star schemas, using special index structure such as Bitmap indexes and Materialized views. Bitmap indexes represent for each key of a dimension table telling which tuples of the fact table have that value. Materialized views answer to general queries from which more specific queries can be answered with less work than if we had to work from the raw data.

ROLAP tools are the traditional SQL-oriented tools that have tight integration to the relational model. The current generations of ROLAP tools are powerful and easy to use. The Relational On-Line Analytical Processing (ROLAP) is emerging as the dominant approach in data warehouse with decision support applications. In order to enhance query performance, the ROLAP approach relies on selecting and materialized in summary tables subsets of aggregate views which are then engaged in speeding up OLAP queries.

4. Creation of Materialized View

In a database management system following the relational model, a view is a virtual table representing the result of a database query. Whenever an ordinary view's table is queried or updated, the DBMS converts these into queries or updates against the underlying base tables. A materialized view takes a different approach in which the query result is cached as a concrete table that may be updated from the original base tables from time to time. This enables much more efficient access, at the cost of some data being potentially outof-date. It is most useful in data warehousing scenarios, where frequent queries of the actual base tables can be extremely expensive.

In addition, because the view is manifested as a real table, anything that can be done to a real table can be done to it, most importantly building indexes on any column, enabling drastic speedups in query time. In a normal view, it's typically only possible to exploit indexes on columns that come directly from (or have a mapping to) indexed columns in the base tables; often this functionality is not offered at all[9].A materialized view definition can include any number of aggregates, as well as any number of joins. In several ways, a materialized view behaves like an index:

- The purpose of a materialized view is to increase request execution performance.
- The existence of a materialized view is transparent to SQL applications, so a DBA can create or drop materialized views at any time without affecting the validity of SQL applications.
- A materialized view consumes storage space.
- The contents of the materialized view must be maintained when the underlying detail tables are modified.

Materialized views are schema objects that can be used to summarize, precompute, replicate, and distribute data. E.g. to construct a data warehouse. There are three types of materialized views:

1. Read only

Cannot be updated and complex materialized views are supported.

2. Updateable

Can be updated even when disconnected from the master site.

Are refreshed on demand.

Consumes fewer resources.

Requires Advanced Replication option to be installed.

3. Writeable

Created with the for update clause.

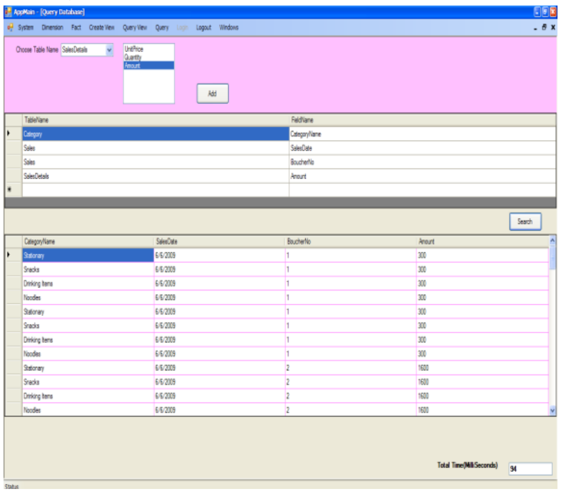
Changes are lost when view is refreshed.

A data warehouse uses multiple materialized views to efficiently process a given set of queries. These views are accessed by read-only queries and need to be maintained after updates to base tables.

A materialized view provides indirect access to table data by storing the results of a query in a separate schema object. Unlike an ordinary view, this does not take up any storage space or contain any data. The existence of a materialized view is transparent to SQL, but when used for query rewrites will improve the performance of SQL execution. An updatable materialized view lets insert, update, and delete. A materialized view can be defined on a base table, partitioned table or view and define indexes on a materialized view. A materialized view can be stored in the same database as its base table(s) or in a different database. Materialized views stored in the same database as their base tables can improve query performance through query rewrites. Query rewrites are particularly useful in a data warehouse environment.

4.1. Query Database Process

In this process, actor choose tables and fields name to view data from database. System shows data according to choose tables and fields name. System shows time span with milliseconds. User can view differences of time span between Query Database Process and Query View Process by viewing two forms.



Choose Table Name: SalesDetails

UnitPrice
Quantity
Amount

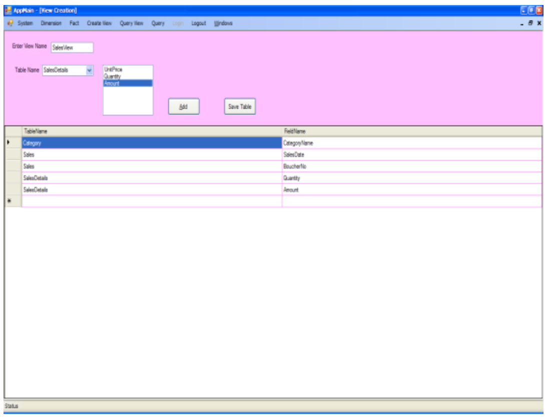
TableName	FieldName
Category	CategoryName
Sales	SalesDate
Sales	BoucherNo
SalesDetails	Amount

CategoryName	SalesDate	BoucherNo	Amount
Stationary	6/6/2009	1	300
Snacks	6/6/2009	1	300
Drinking Items	6/6/2009	1	300
Noodles	6/6/2009	1	300
Stationary	6/6/2009	1	300
Snacks	6/6/2009	1	300
Drinking Items	6/6/2009	1	300
Noodles	6/6/2009	1	300
Stationary	6/6/2009	2	1600
Snacks	6/6/2009	2	1600
Drinking Items	6/6/2009	2	1600
Noodles	6/6/2009	2	1600

Total Time(MilliSeconds) 94

4.2. Materialized View Creation

This is the materialized view process using Structured Query Language (SQL) statements. In this form, actor choose fields name from dimension and fact tables to create view table. And entry view name. System auto create view table and save data according to choose fields and tables.


AppMain - [View Creation]
System Dimension Fact Create View Query View Query Login Logout Windows
Add
Save Table
Status

Enter View Name: SalesView

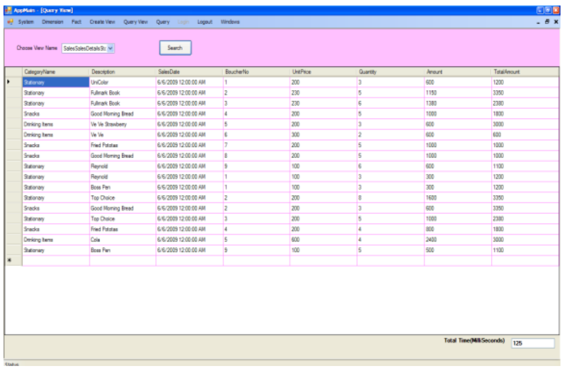
Table Name: SalesDetails

UnitPrice
Quantity
Amount

	TableName	FieldName
▶	Category	CategoryName
	Sales	SalesDate
	Sales	BoucherNo
	SalesDetails	Quantity
	SalesDetails	Amount
*		

4.3. Query View Process

In this form, system shows data according to choose view name by using SQL statement. System shows timespan with milliseconds.


AppMain - [Query View]
System Dimension Fact Create View Query View Query Login Logout Windows
Search
Status

Choose View Name: SalesSalesDetailsSt

	CategoryName	Description	SalesDate	BoucherNo	UnitPrice	Quantity	Amount	TotalAmount
▶	Stationary	UniColor	6/6/2009 12:00:00 AM	1	200	3	600	1200
	Stationary	Fullmark Book	6/6/2009 12:00:00 AM	2	230	5	1150	3350
	Stationary	Fullmark Book	6/6/2009 12:00:00 AM	3	230	6	1380	2380
	Snacks	Good Morning Bread	6/6/2009 12:00:00 AM	4	200	5	1000	1800
	Drinking Items	Ve Ve Strawberry	6/6/2009 12:00:00 AM	5	200	3	600	3000
	Drinking Items	Ve Ve	6/6/2009 12:00:00 AM	6	300	2	600	600
	Snacks	Fried Potatas	6/6/2009 12:00:00 AM	7	200	5	1000	1000
	Snacks	Good Morning Bread	6/6/2009 12:00:00 AM	8	200	5	1000	1000
	Stationary	Reynold	6/6/2009 12:00:00 AM	9	100	6	600	1100
	Stationary	Reynold	6/6/2009 12:00:00 AM	1	100	3	300	1200
	Stationary	Boss Pen	6/6/2009 12:00:00 AM	1	100	3	300	1200
	Stationary	Top Choice	6/6/2009 12:00:00 AM	2	200	8	1600	3350
	Snacks	Good Morning Bread	6/6/2009 12:00:00 AM	2	200	3	600	3350
	Stationary	Top Choice	6/6/2009 12:00:00 AM	3	200	5	1000	2380
	Snacks	Fried Potatas	6/6/2009 12:00:00 AM	4	200	4	800	1800
	Drinking Items	Cola	6/6/2009 12:00:00 AM	5	600	4	2400	3000
	Stationary	Boss Pen	6/6/2009 12:00:00 AM	9	100	5	500	1100
*								

Total Time(MilliSeconds): 125

5. System Design

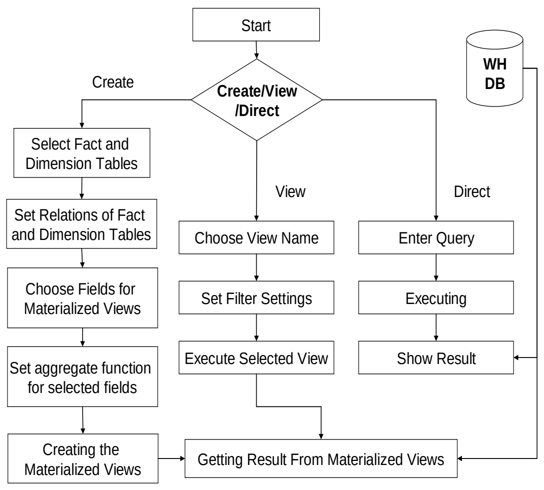

Start
Create/View /Direct
Create
View
Direct
WH DB
Select Fact and Dimension Tables
Set Relations of Fact and Dimension Tables
Choose Fields for Materialized Views
Set aggregate function for selected fields
Creating the Materialized Views
Choose View Name
Set Filter Settings
Execute Selected View
Enter Query
Executing
Show Result
Getting Result From Materialized Views

Figure1. System Flow Diagram

In this system has three parts, view creation, extraction of data using view and from base tables. View creation can be done by administrator. To create view, firstly select fact and dimension tables and then set relation of these tables and choose fields for materialized view. Then set aggregate function for selected fields and materialized view had been created. Extraction of data directly from views and original base tables can be done by any users.

6. Conclusion

This paper can create materialized view tables according to choose tables and fields name. System will construct SQL statement automatically to create view tables and to bind data to view tables. By using materialized view creation, user can view data from view tables with less time. So to view fast data, view materialization is a viable alternative. Precomputing aggregates on some subsets of dimensions and their corresponding hierarchies can substantially reduce the response time of a query. OLAP applications use precomputation of aggregate data to improve query response time.

Materialized Views can reduce system cpu/io resource requirements by pre-calculating and storing results of intensive queries and allow for the automatic rewriting of intensive queries can be refreshed on demand or on a schedule. Materialized views have been found to be very effective at speeding up queries, and are increasingly being supported by commercial databases and data warehouse systems. However, whereas the amount of data entering a warehouse and the number of materialized views are rapidly increasing, the time window available for maintaining materialized views is shrinking. These trends necessitate efficient techniques for the maintenance of materialized views. In this paper, we show how to find an efficient plan for the maintenance of a set of materialized views, by exploiting common sub expressions between different view maintenance expressions.

7. References

[1] A.Savagaonkar and S.Kulkami,"Materialized View Definition and Maintance", Indian Institute of Technology, Bombay.

[2] E.K.Sze and T.W.Ling," Materialized View Maintenance Using Version Numbers", School of Computing, National University of Singapore, Lower Kent Ridge Road, Singapore 119260.

[3] G.Limaye,"Data Warehousing with Materialized Views", Release 8.1.5, A67775-01.

[4] N.Roussopoulos and Y.Kotidis,”An Alternative Storage Organization for ROLAP Aggregate Views Based on Cube trees”, Department of Computer Science, University of Maryland, kotidis@cs.umd.edu, nick@cs,umd,edu.

[5] S.Agrawal and V.Narasayya,”Automated Selection of Materialized Views and Indexes for SQL Databases”, Microsoft Research, sagrawal@microsoft.com, vivdknar@microsoft.com.

[6] S.Chaundhuri and U.Dayal "An Overview of Data Warehousing and OLAPTechnology", Microsoft Research,Redmond,urajitc@gmail.com,dayal@hpl.hp.com .

[7] A.Berson and S.J.Smith,”Data warehousing, Data Mining and OLAP, ISBN-0-07-006272-2.

[8] D.M.S.Anahory,”Data Warehousing in the Real World”, ISBN-0-201-17519-3.

[9] http://en.wikipedia.org/wiki/Materialized View.

[10] http://www.ss64.com/orasyntax/3views,html.

[11] http://youngcow.net/doc/oracle10g/server.102/b14 200/statements 6002.htm.

[12] http;//en.wikipedia.org/wiki/Data_warehouse.