Datawarehousing : Future Direction

*Dr S Chandrasekhar

ABSTRACT

Global economy today demands that organizations need to change constantly to changes in environment to remain competitive. To do that one has to capture data and convert the data to information which will be an input to decision making. Data comes from different sources, both from within and external to the organization. They are generated from different application systems. The formats will be different, timings will be different. There is a need to integrate, summarize and move the data throughout the organization. This is the main purpose of the data warehouse. It will integrate/summarize and allow access to data in a manner which can be easily understood by decision makers hiding the technical complexities. The warehouse also stores quite a good volume of historical data so that trends and patterns can be identified from the past and projected to the future. It also allows the data to be explored in an unconventional way rather than sticking to a rigid framework.

Since the data warehouse stores large volume of historical information and allows both structured and unstructured queries, it poses lot of challenges to the way data has to be organized, accessed and presented. This article discusses some of the issues and challenges to accomplish this task and indicates a future road map to the developments that are likely to take place in this field.

Introduction

As explained above because of the nature of application which a data warehouse handles is different from normal database applications, the structure of the data is different. Normally in a typical database application one expects a very fast response to a query and to achieve this objective, the data is scattered in different tables. At the time of query these tables will be joined to produce the desired output. Most of the time database queries are known before hand and keeping this in mind the optimal table design can be achieved.

Data warehouse stores large volume of historical data and in a majority of cases summarized data is also stored. Data is organized among business dimensions like time, location, product etc. Nature of queries is also complex when compared to a RDBMS QUERY. It should also allow the users to access the data in a way that they would like to do rather than in a predefined framework. Also one should be able to identify the trends/patterns from historical data. Response to the query need not be that fast compared to RDBMS application.

To achieve these objectives invariably data needs to be kept in a demoralized format with as few tables as possible. Commonly used data storage structures in a data warehouse are:

* Star schema
* Snowflake Schema: A variant of star schema
* Multi dimensional Databases (MDDB)

*B.Tech, M.Tech. (IIT, Kanpur) Ph.D (USA), Chair Professsor & Director -IT FORE School of Management, New Delhi.
e-mail: sch@fsm.ac.in

* Relational structure with conversion to MDDB on fly

Data from operational systems needs to be loaded to the warehouse. This process is called Extraction, Transformation and Load (ETL). Normally data from the operational systems goes to a staging area, in the staging area data cleaning, Transformation (if required) will be carried out and this cleaned data will be propagated to the Data warehouse. The tool that performs this task is known as ETL. ETL tools also generate the metadata which is used to access the Data warehouse. To access the dataware we use front end tools that are known as On line Analytic Processing (OLAP).

Issues With The Storage Strucrtures Of The Warehouse:

In Star Schema there will be a central fact table and few dimension tables connected to the fact table. What parameters one would like to monitor will be kept in the fact table and those attributes along which one would like to visualize the facts will be made dimensions. As a simple example if one has two entries in the fact table say sales and margin and in Dimension say we have location (5), Product (50) and Time (12) Quarters of data i.e. 3 years), then the number of fact table rows will be 5 x 50 x 12 = 3000. Apart from this few more summary tables will also be stored so as to speed up the query.

In Snowflake Schema (which is a variant of STAR) Dimension table will be split to an additional level, otherwise it is like STAR schema.

MDDB is conceptually like a Ruby Cube. Dimensions will be the axis of the cube and the contents of the cubes will be facts. To process a Query, relevant cubes which lie at the intersection of the query, will be extracted for processing. MDDB is stored as Arrays.

As the number of users using the data warehouse grows the amount of data increases, the size of Fact tables, MDDB grows in an exponential way and performance becomes a concern. We end up having Fact table with millions of rows, MDDB's with large number of cubes. In fact MDDB access becomes a major bottleneck when the Dimensions exceed 5 and size goes beyond 50 GB. Large fact table will reduce the Query response time.

Some Of Options To Dw Architecture

First Alternative

Run the ETL Process, physical warehouse and Access (OLAP) on Separate Servers.

Up to certain size of the Data warehouse and Number of users this may solve the problem. But as the size of DW grows this may not be enough.

Second Alternative(s)

The second alternative is to increase the speed of data access. This can be done by:

* Hardware options
* Specialized software with manipulation of Index tables rather than the Data.

Hardware options:

This includes faster disk drives and multiple CPU's. Faster disk drives help by increasing the I/O rate but still cannot compensate for the massive amount of Data. Multiprocessors provide enormous speed advantage but as long as data has been partitioned on Multiple Disk drives more or less in a uniform way. The issue is to partition the data on to multiple disks. Partitioning provides better data manageability, quicker back up and fewer table reorganizations. Table partitioning is done by using Data element keys, which can leverage multiple processes running in different CPU's in parallel.

Partitioning keys are used to separate the data into physical Separate data sets. The most common keys can be associated to Time, Location, and Product/Customer id. etc. Each key combination needs to be carefully examined so as to balance the number of rows in each partition. Additional Consideration needs to be given on how the data will flow over the lifetime of the system.

Three commonly used methods to create positions are:-

* Associating partitioning keys to Transaction time
* Associating partitioning keys to value
* Associating partitioning keys to date.

Manipulation of Index Tables

When the data in the Data warehouse is kept in star/ snowflake schema, Database Administrators need to create countless indexes and summary tables in order to optimize the performance for known queries. But when user queries the DW in a new way this method takes time as indexes and summaries have to be generated afresh. The solution is to access indexes and perform counts and aggregations at Index levels only without accessing the underlying database. This reduces the disk I/O to a considerable extent.

According to Bill Inmon, it is more efficient to service a Query by simply looking in Index instead of going to main data. But many cannot do this. The Solution is to build an intelligent index processing capability.

When a multidimensional query is requested, the required values are accessed within the Indexes & the results joined at Index level. They are optimized for cross table retrievals. Since Indexes are manipulated the size of the DW will have almost no impact on the response time.

The same Index access technique can be used for Data Aggregation also. A data aggregation is usually a COUNT, SUM, MIN, MAX OR AVERAGE grouped by another column. Such Aggregations can be created using on Aggregation Index. Once the data is selected through multidimensional indexes, this information will be passed on to Aggregation Index to process the desired output.

In short Index manipulation methodology enhances the response of DW Query by:

* Reduced need for summary tables
* Any level of Granularity
* Access to detail data if required.

Conclusions

Data warehouses are by far the largest and most computationally intense business applications at most enterprise. They consist of huge time history databases spanning hundreds of Gigabytes of data. As the DW size grows in size to handle reasonable number of user's innovations in Data Access and hardware architecture, scalability becomes an important issue. Also the technique of Simulation & Data mining will become a part of OLAP/DW Engine so that selection of partitioning keys, Index Manipulations & moving the less used data to a cheaper storage will be automated rather than to depend upon the expertise of a DBA. These things will guide the Data Base Administrator in decision making for optimum utilization of resources. The article gives an insight into some of the issues that will come up in near future.

References

Inmon W.H. "*Data warehousing Architecture and Implementation*" Prentice Hall, 1999

Inmon W.H. "*Data warehouse Performance*" Johniley, NY, 1999

Meyer, Donetal *"Building a better data warehouse"* Prentice Hall, 1998

Paulray. P. *"Data warehousing fundamentals"* Wiley, 2003

Thomsen, *Erik "OLAP Solutions: Building Multidimensional Information Systems"*, Wiley, 1997

www.datamirror.com
www.dmreview.com
www.disc.com
www.ncr.com