Data Warehouse Architecture: A Review

Dipra Mitra
Technique Polytechnic Institute
Panchrokhi, sugandiya
District: Hooghly, pincode: 712102
(West side of Hooghly rail station of howrah-bandel line), West bengal, india

Abstract
Data ware housing is very efficient to analyze Mass data and it helps to do decision making. This paper introduces architecture of data ware housing and some model like star schema for analyzing the mass data.

Keywords - data ware house architecture, star schema.

I. Introduction
In modern computing data ware housing is a system for reporting and mass data analysis it is used as central repositories of integrated historical data from one or more displaced sources.

II. Data Warehouse Architecture
The basic architecture for a data warehouse environment is shown below: The data warehouse is stocked by a variety of source databases from possibly different geographical locations. Each source database serves its own applications, and the data warehouse serves a Decision Support System (DSS) or a Executive Information System(EIS) with its informational requests. DSSs are used to analyze data from commonly available databases with multiple sources to create reports. The report data is not time critical in the sense of real time systems but must be timely for decision making. EISs are like DSSs but more powerful, easier to use and more business specific. Each feeder system Database must be reconciled with the data warehouse data model, this is accomplished in the processing that takes place for the extraction of the data from the feeder database system, the transformation of data from the feeder system to the data warehouse, and the actual loading of data into the data warehouse.

In a two-tier architecture (below), Tier 1 is a collection of back-end servers for specific applications, all connected to a backbone network, and tier 2 is a collection of mainframe clients with their own databases and access to the same backbone network. The mainframe clients (legacy systems) are left alone but have special front-end systems to provide transparent access to legacy systems. This approach does not scale well to more systems or to large numbers of on-line users.
A three-tier architecture (shown below), works better than the 2-tier systems by putting the clients at tier 3, the local servers and graphically oriented user interfaces at Tier 1, and the high speed database and application servers at Tier 2, which handle business rules and data that are shared across the organization.

The Tier 2 servers provide high-speed access to the shared data in the data warehouse. Typical middleware used to realize Tier 2 are CORBA, Microsoft’s DCOM, Sun’s RMI, and Java Beans. The choice of the best architecture depends on the size of the data warehouse and the expected traffic. The Three-tiered architecture tends to be more expensive but more easily scalable for the larger systems.

III. Star-Schema

The dominating conceptual data model for data warehouses is the multi-dimensional view, based on 2 factors: a set of mostly numeric measures that define the objects associated with the subject areas (e.g. sales-total, customer-id, order-no) and a set of dimensions (represented by entities) that provide the context for the measures (e.g. products, orders, regions, customers). A collection of dimensions uniquely identifies each measure, and each dimension is described by either a set of flat attributes or a hierarchy of attributes. If we choose to use existing relational technology, we can implement the multidimensional view using star schema.

The star schema consists of a single fact table and a set of dimension tables, with a single table for each dimension. The fact table contains the numeric or nonnumeric measures described above and has the basic transaction level data for the business. The fact table is usually extremely large; it has numerous rows due to the many possible combinations of dimension values and a number of columns equal to the number of dimensions it represents simultaneously.

The dimension tables are usually much smaller, containing the largely non-numeric data associated with the attributes of the dimensions (or entities). The fact table acts like an intersection table for a many-to-many relationship or ternary relationship for the pure ER model, in that it tends to contain only the keys of the related entities (dimension tables) as its foreign keys. In effect the fact table is “many” side of a one-to-many relationship with each of the dimension tables. The connection of a primary key in the dimension table and its corresponding foreign key in the fact table allows this approach to provide referential integrity to data.

The major advantages of the star schema are performance and ease of use. The star schema allows for a variety of queries to be effectively processed, and it is intuitive for most end users to visualize. A typical star schema for an “order” database is shown below:
The fact table is created to connect dimension tables through the concept of an order. The dimension tables are created for the dimensions order, product, salesperson, customer and region.

As an example of the performance benefit of the star schema we consider the query: “rank the regions in terms of number of units of product x sold” using the above figure. First we look up product x in the product dimension table to get the prod-no value. Then we effectively join the product table with the fact table for product x, using a fast (bit-map) indexing technique to avoid a full table scan of the fact table. For each record found in the fact table, the quantity of product x sold is added to the subtotal for the region number in that record. If there are 500 products, only 1/500th of the fact table on the average needs to be accessed to answer the query. Similarly, the time dimension added would allow us to answer queries with time constraints quickly without having to scan all the data over all the time periods. Also, dimension tables may be unnormalized. For instance, if the salesperson’s address determined the phone number, or the actual address included street number, city, state, and zip code where the zip code is dependent on the other components, then that fact table could be considered unnormalized.

IV Conclusion:

It can be concluded that this architecture provides how historical data has been arranged through different tiers and how it can be implemented for accessing historical data from a mass data

Reference:


