LECTURE UNIT ON DATA WAREHOUSING WITH A CASE STUDY

LECTURE UNIT ON DATA WAREHOUSING WITH A CASE STUDY

By

KEDONG LIN, B.Eng.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

McMaster University

MASTER OF SCIENCE (2007) **McMaster University**

(Computing and Software) **Hamilton, Ontario**

TITLE: **Lecture Unit on Data Warehousing with a Case Study**

AUTHOR: **Kedong Lin, B.Eng. (Shanghai University)**

SUPERVISOR: **Professor Frantisek (Franya) Franek**

NUMBER OF PAGES: **xii, 118**

Abstract

Data warehousing can be considered as a relatively new and still evolving technology in IT industry. The term *data warehouse* was coined by Bill Inmon in the early 1990's and he defined it as a subject-oriented, integrated, time-variant, and nonvolatile collection of data for management's decision support systems. The main task of a *data warehousing system* is to retrieve and integrate data from distributed and heterogeneous information sources, and load the summarized data into a data warehouse for further business analysis.

Data warehousing can thus be considered the infrastructure part of the decision support system (DSS) - it supports the information management for business decision making and enables the online analytical processing (OLAP).

Date warehousing is not a simple system for small scale businesses - a typical data warehousing system costs on average more than $3 millions of dollars a year for hardware, services, and system integration, according to a survey performed in 1996. On the other hand, a successful data warehousing project can bring in excess of 500% in return on investment. As a result, data warehousing has become a major component of business systems since its conception. Although the technology has more than 15 years of history, new upgrades and products for data warehousing still appear every year.

Most people interested in data warehousing can be found in business environments. The educational institutions just began to embark on data warehousing projects only recently. The goal of my thesis is to describe concepts and methods of data warehousing systems in a form

of a specialized unit within an undergraduate database course. In particular I focus on data warehouse system architecture, data models, online analytical processing (OLAP), and warehouse data processing.

Finally, a case study is included in this thesis to help understand the implementation of a data warehousing system in the commercial environment.

Acknowledgements

First of all, I would like to thank Dr. Frantisek (Franya) Franek, my supervisor, not only for his guidance and help during this research, but also his kind support during my graduate studies.

I must present my great thanks to my family - my parents, sister, and brother-in-law. Although they are not in Canada, they have given me great and unequivocal moral support. As a part-time student, I have had to reconcile both, the graduate studies and a very demanding job as a Database Administrator. Frequently, I have had to study late at night. It is extremely hard to travel between Toronto and Hamilton, especially during the winter. Without their encouragement, I would have given up in the first term of my studies.

Sun Li, my previous colleague in the DBA team, has given me a lot of advice for this thesis. His highly practical experience with data warehousing comes from many data warehouse projects he had participated in. I am very grateful for his help.

Finally, I wish to thank to Yuemin Zhu who has given me a lot of support during my studies.

Toronto, Ontario — Kedong Lin

November, 2006

To my parents and sister.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background and History

1.1.1 Decision Support Systems and Their Shortcomings

In today's global economy, most companies face rapid changes in the business environment compounded by a heighten competition. Managers have to adjust the enterprise policies and goals constantly to meet the new challenges. Thus, the information systems are ever so more important in business operations. They help the management plan for the future, increase the enterprise efficiency, and enhance the company's competitive capabilities. Various information systems, like Executive Information Systems (EIS) or Decision Support Systems (DSS), have been established for such purposes.

Many definitions have been offered since the term "Decision Support System" appeared in 1970s. It can be best summarized as *a specific computerized information system offering business planning and organizational decision-making.* As an

interactive software-based system, the goal of DSS is to assist the decision makers by extracting relevant business information from the raw data for the purpose of business analysis to chart the appropriate business strategy.

If the current corporate data are integrated with historical data and both made accessible, EIS and DSS can still be effective. Unfortunately, most of EIS and DSS do not have such an ideal environment since legacy systems and operational databases cannot meet these requirements. The operational information systems are very diverse dealing with different business objects and different transaction activities. Therefore, it is hard to build an enterprise-level DSS because of the incompatibilities of various operational systems that hamper daily operations and decision making performed at the same time.

The properties of current operational database systems prevent the integration of decision-support functionality into the database systems directly:

1. A typical operational database system contains a lot of very detailed data. In general, DSS will not analyze those detailed data directly. Before any analysis, we need to integrate and aggregate the data according to specific business requirements. The summarized data are the fundamental precursor for an effective analysis and decision making in DSS. In practice, most business data to be considered are OLTP (online transaction processing) data and as such rather diffused. If we were to use the original data for analysis as they are, the analysis application would need a lot of extra modules to aggregate and integrate the raw data before it could generate any useful analysis. The aggregation and integration of data would slow significantly the analysis and would be repeated any time a new analysis is requested.

2. Furthermore, we may need to access various databases on various platforms to collect the required information. Obviously, the data from different systems may not abide by the same standard format. This would cause conflicts and confusions. In reality, it is beyond the users ability to assemble useful data for analysis purpose from many diverse sources. In addition to various formats, the data from various sources are probably not synchronized. For example, in one system a particular value has been updated while in another system not yet.

3. Due to the complexity of required analysis, we could not base the analysis on the data from operational database systems. For instance, OLTP data are designed to reduce data redundancy and improve the transaction system performance and not to facilitate business analysis.

4. From many data sources, we can only obtain current data values, but cannot obtain historical data. For instance, we can find the current product price, but not the prices for the past five years.

5. Often the main role of the operational database system is to support OLTP. For decision support we need an environment that allows OLAP (online analytical processing). These two activities exhibit contradicting processing requirements – in an OLTP environment, the data are accessed frequently, but the access time is short, while in an OLAP environment, the analysis process will not touch data as often as OLTP, but some complex and historic analysis will hold system resources for a couple of hours or even longer. If the decision support system shares the database system with an OLTP system, it will bring serious performance issue to daily operational users, and for an operational system,

response time is always mission critical.

Due to the above reasons, a new technology - data warehouse, is needed as a fundamental component of a decision making system. A data warehousing system translates simple data into business information and knowledge.

1.1.2 Data Warehouse Overview and Evolution

The concept of data warehousing appeared in the early 1990s. Bill Inmon, who is considered as the father of data warehousing, defines data warehouse as a subject oriented, integrated, non-volatile, time variant, summarized and detailed collection of data supporting management's decision making for the entire organization [3].

First, we will discuss the evolution of data warehousing since 1970s. A few decades ago, in order to provide some business analysis, people had to use the operational database with some additional modules. The decision making system was based on one or multiple operational databases or other data resources.

This was the conception of the idea that a **virtual data warehouse** is needed for the analysis function of the decision support system. The virtual data warehouse would consist of one or several data resources, databases or flat files. However, such a system could satisfy some simple requirements for analysis, but hardly any complicated and sophisticated business analysis due to the diverse nature of the data in such an environment – the data are inconsistent, incomplete, and not uniform as required for a more complex analysis. Furthermore, historical data are often not available in databases devoted to OLTP, and the analysis of historical data is an essential component for long-term analysis and decision making. Naturally, the operational data is a major asset to any organization, but we cannot truly utilize its full potential without

first "refining" it. With refined data, companies can quickly identify the business trends and develop the coping strategies.

After the 1980's, a new technology of so-called **data extracting** became popular. Data extracting searches databases and/or data files, selects some data meeting pre-specified criteria and converts them into a specific data file or database. It combines and integrates some useful data and generates new knowledge from the raw data.

There are three reasons why data extracting became quite popular:

1. Data extracting separates data for analysis from the operational database, so the time and resources consuming analysis process is not in conflict with the online transactional process.

2. Once the data for analysis are moved out of operational database or legacy system, the ownership and control are shifted.

3. Through processing of these rudimentary data with certain complex analytical techniques, people can obtain new and valuable knowledge about the company operation.

From the 1990s, people tried to consolidate all the related data from many different systems and sources, and store them into a separate data warehouse for business analysis. A data warehouse contains a huge amount of integrated data from all heterogeneous systems which constitute the major areas of an organization. Data warehouses filter, integrate, and keep all historical data. Usually, data warehouse is a read-only database system to assure persistency of the data. More and more organizations use such a data warehouse system for all decisional and analytical modules, instead of accessing different operational data sources. Data warehouse is one-step

store which collects all necessary information. Moreover, data warehouse can be considered as a forefront technology in IT industry today. It provides a fundamental tool to make more effective and accurate business planning and decision.

As mentioned above, the main purpose of data warehousing is to collect all useful data into a separate specialized storage location for convenient access. It should be mentioned that besides the *centralized data warehouse* approach, there is a *distributed data warehouse* architecture. In the distributed data warehouse architecture, similarly as with distributed databases, the data warehouse is separated into several sites. However, the distributed data warehouse architecture is not as popular as the centralized approach, but may be suitable for certain environments. A discussion of the advantages or disadvantages of the distributed architecture and the conditions suitable for the distributed approach, though interesting, is outside of the scope of this thesis.

1.2 Data Warehouse Objectives

Data warehousing has become the most effective solution in data optimizing and business analysis since the 1990s. By using a data warehouse system, it is possible to gather all the needed data into an optimized database, no matter how different and diverse the data sources and data applications are. In the past 10 years, the data warehouse technology has been implemented in most of the top companies in the world. However, a data warehousing system is still relatively expensive, including the hardware, tools, services, and system integration. An ETL (Extract, Transform and Load) tool alone can cost more than $40,000.00 and it is just a single component of a data warehousing system. According to some research, the minimum cost of a

small data warehousing system is estimated about $100,000.00 . It is thus clear, that the objectives of a data warehouse system must be to provide a high-value service to justify the costs.

The fundamental goal of a data warehouse is to provide information for decision making and OLAP application at the enterprise level. In order to access information from a homogenized and comprehensive view of the organization, the data in data warehouse are organized by subjects to support applications such as comparisons, trend analysis, and business forecasting.

Listed below are several major activities of a data warehouse system:

1. Constructing an environment containing all necessary data ready for further analysis.

2. Integrating data from various heterogeneous sources and unifying different data formats.

3. Building a collection of consistent historical data.

4. Providing a support for business analysts to make beneficial business decisions for organization.

5. Satisfying various additional demands of the end-users.

1.3 Important Factors for Implementation of a Successful Data Warehouse

Most of the companies in the world today will consider implementing a data warehouse since it consolidates useful information to facilitate future business decision making.

This technology has already been used in a number of industries including health care, finances, manufacturing, services, transportation, and telecommunication.

However, the failure rate of data warehousing projects is higher than 50% according to IDC research in 1996 [1]. (IDC is an USA company specialized in providing market intelligence and advisory services concerning the information technology, telecommunications, and consumer markets.) These failures do not include those systems that the end-users are not completely satisfied with. At the same time, on average a practical data warehouse project costs $3.2 million and 3 years of development time. What is worse is that most project managers for data warehousing find the problems they focused on 3 years ago are no longer the same today. The project failures are related to data sources, funding, resources, time, hardware and software required to sustain such a project. Other leading reasons to cause failures in data warehouse projects are many complex technologies, unclear requirements, unrealistic objectives, poor performance, and poor data quality.

The factors for the success of a data warehouse project have to be considered in advance. According to Haley [2], the most important factors are the following:

1. Project planning.

   Typically a data warehouse project will last an extended period of time. Good project planning and management are critical factors for a successful data warehouse system. Similar to all other projects, a data warehouse project management involves project integration management, time management, cost management, quality assurance, human resource management, communication management, and risk management. The objectives of the data warehouse

system should be clarified before the project starts. Metadata library and software/hardware selection criteria need to be included.

2. Prototyping

   Prototyping is a communication tools between business and end-users. It builds the basis for the final working system.

3. User participation

   Naturally, data warehouse project developers need to collaborate with the operational system staff to get the job done. However, final users also need to be involved in the project from the beginning. It helps to pinpoint and satisfy the requirements of the end-users and build a successful data warehousing system.

4. Having the right skills

   The skills required from the project team members involve both technical skill and business knowledge.

5. Quality of the data sources

   The goal of a data warehouse is to provide structured and integrated data to assist the business decision making of an organization. Hence, the data should be accurate, consistent, and complete. Data source quality is only the first step to warrant the final information accuracy.

6. Simple technology

   User-friendliness always should be a factor considered in advance for every project. End-user will not be satisfied with any tools that are difficult to use.

Chapter 2

Data Warehousing System

2.1 Definition

Ralph Kimball, a noted authority on data warehousing, defines data warehouse simply as a copy of transaction data specifically structured for query and analysis [10]. Of course, there are some other definitions – for instance – a data warehouse is a consolidated database, which contains a huge amount of integrated data organized around major subject areas of an organization that span over a period of time to serve a historic purpose [5].

All definitions thus define a data warehouse as a collection of data separate from the operational data source for the purpose of supporting an organization business strategy and planning. Most researchers agree that data warehousing system is a fresh concept, but not a really new product. Because data warehouse is specifically designed for business analysis, decision making and future planning, system design and data model are quite different between data warehouse and operational database. Both system architecture and data model try to provide a better service for decision making and analytical processing.

Chamoni & Gluchowski [6] describe the relationship between data warehouse and

operational system using the diagram in Figure 2.1:

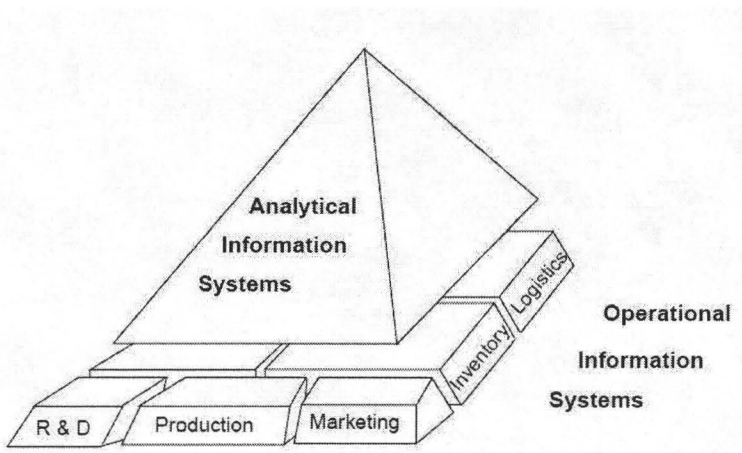


Figure 2.1: Enterprise Information System

2.2 Data Warehouse Architecture

The data warehousing systems and architectures vary in different organizations. However, there are three basic layers in a data warehouse system: data source, data warehousing database, and end-user application.

There are also basic components that we find in any data warehouse system.

1. Data Source

   The data typically comes from relational databases such as Oracle, Informix, Sybase. or DB2. There are some other data sources for a data warehouse, like flat data files or any other external data.

2. ETL component

   Extracting, Transforming, and Loading of data is a very important activity of a

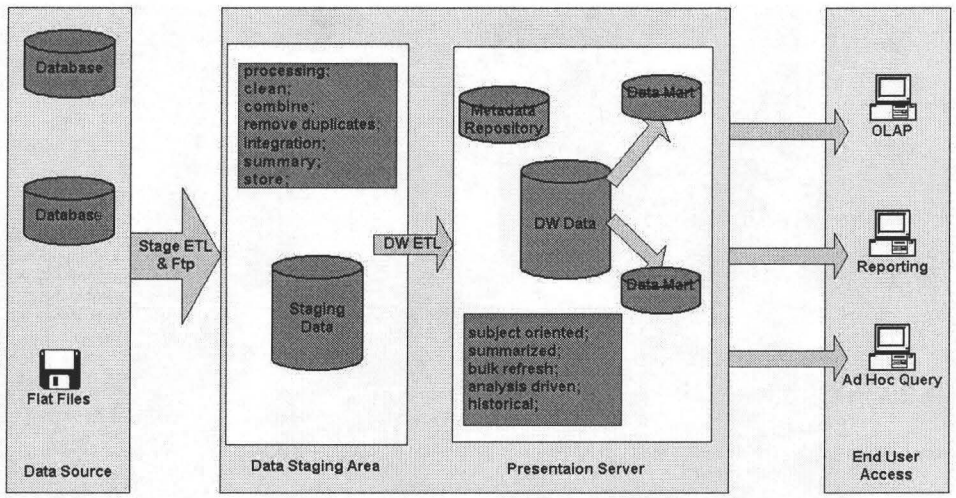


Figure 2.2: Data Warehousing Architecture

data warehousing system. The complex process helps improve data quality and organize data for analysis purpose. It extracts, transforms and loads data from all data sources to data warehousing database. ETL component also performs initial data load and incremental refresh load.

There are two types of ETL: Stage ETL and DataWarehous ETL. Stage ETL collects the original transactional records and move them into data staging area.

analysis processing. Metadata serves also as a manual for the future maintenance.

cleaning system between data source and presentation server. Data staging area can also be considered as a construction place for the data warehouse. Data staging area stores all detailed data. The detailed data will be regularly added to support summarized data. A set of processes to perform these operations has to be implemented in this area. However, data in data staging area will not be available online before they are transformed onto the presentation server.

4. Presentation Server

   Presentation server is the physical target database. After a series of data processes, data warehousing data is loaded for end-users analytical processing. End-users will only access the presentation server. Data model used in the presentation server varies depending on the specifics of the organization. The most frequently used data models for data warehousing are relational and dimensional data models.

5. DataWarehouse Schema Scripts

   These scripts assist the DBA (Database Administrator) to recreate an empty data warehouse and support objects like indexes and views. Shell scripts and scheduled jobs to automate the initial load and he incremental refresh process also belong to this software collection.

6. Metadata

   It is very important for a data warehousing system to keep all metadata (data about data). Metadata is used to map data source to a data warehouse schema during the development. It also helps end-users understand and implement the

analysis processing. Metadata serves also as a manual for the future maintenance.

7. End-user layer

   The main task of data warehousing is to provide information for business analysis. The analytical functions are implemented in the end-user layer. There several modules or tools to interact with the data warehouse at this layer; typically application development module, query module, reporting tools, OLAP tools, and data mining tools.

2.3 Data Warehouse Characteristics

In contrast to an operational database, a data warehousing system is characterized by special features.

2.3.1 Data in a Data Warehouse are Subject-Oriented

Unlike the OLTP system where data is designed for application, data warehousing system is organized by subjects such as product, sales and customer, etc. For example, an insurance company's OLTP system data model probably will be designed around its different products, such as auto, home and life insurance policies. In contrast, the same company's data warehousing system will arrange data according to subjects such as claim or policy. The subject-oriented characteristic thus affects the design and implementation of a data warehouse.

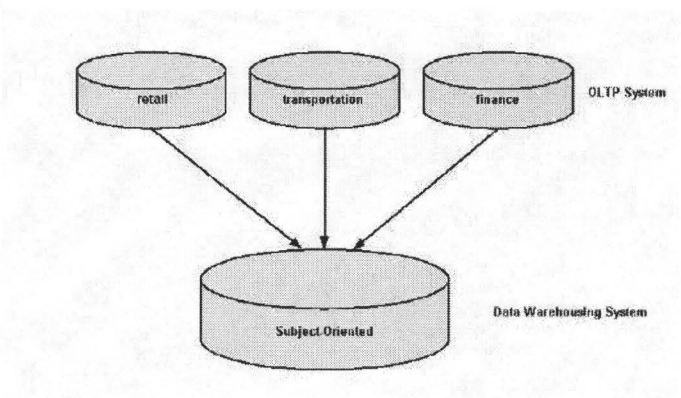


Figure 2.3: Subject-Oriented Information

2.3.2 Data in a Data Warehouse are Integrated

The data for a data warehouse typically come from many different sources. Conflicting and inconsistent data must be removed before the data are loaded onto the presentation server. The implementation process involves enforcing *naming consistency*, *format consistency*, and *data consistency*. A global schema is built according to all source data.

For instance, two banks, A & B, just merge their businesses. However, two banking systems use different formats for account number. Bank A's system account code consists of a numeric 15-digit code while Bank B's system account code uses 3 alphanumerics + 10 numeric digit. Furthermore, Bank A uses "account number" as the column name but Bank B uses "ACT" to represent the account. To build a subject-oriented account object for a data warehousing project, a column name needs to be used and the different source data must be integrated. There is no need to change the coding in these original systems, but we must create a mechanism to change the incoming data and define a common account schema.

2.3.3 Data in a Data Warehouse are Time-Variant

Unlike the operational databases which only keep the current values, data warehouse needs to keep a few years of data for decision making and trend forecasting. Historical data is useful for consistent comparisons. In most cases, more than 10 years of data is usually stored in a data warehousing system. Time variance is definitely a major feature of a data warehousing system. Basically, a time stamp needs to be implemented for the records. Data in a data warehouse are never updated, only be appended.

2.3.4 Data in a Data Warehouse are Nonvolatile

Insertion, deletion, and modification are normal operations for traditional databases. However, data for a data warehouse is bulk-loaded from all data sources periodically and then normally will not be changed anymore after data loading.

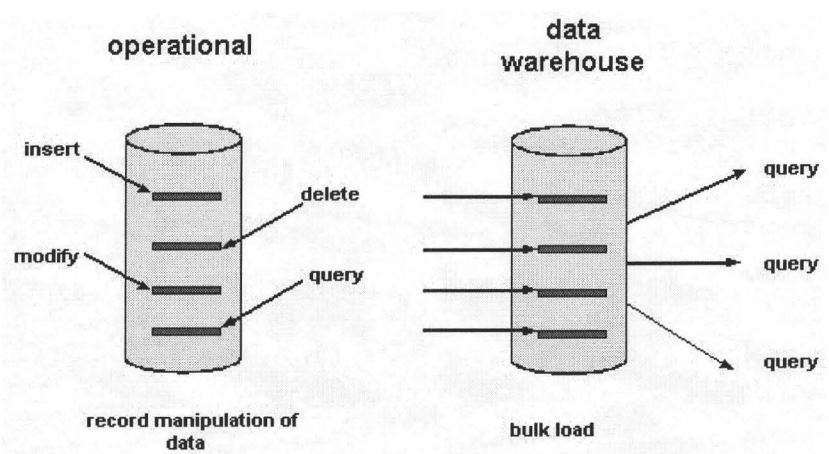


Figure 2.4: Database (Read/Write) vs. Data Warehousing (Read Only)

2.3.5 Data in a Data Warehouse form a Collection

Data warehouse data is a collection of all kinds of data from various resources. However, they are integrate, summarized and mapped into usable form for business planing and analysis.

2.4 Comparing Operational Databases and Data Warehouses

A database is, simply put, a computerized record-keeping system. As such, the purpose of a database is to store a collection of logically related data as a single cohesive unit [7].

Operational system is typically designed to accurately document detailed daily business operations of an organization. The goal of an operational database is set according to the context of the company's business activities.

The data warehouse differs from the operational database; it is oriented towards some important subjects for company decisions such as products, customers, business, activities or policies. Data warehouse is used to facilitate analytical applications, assist business decisions and business forecast.

Operational database is to serve business transactions. Most of time it is used in a real-time system to automate collecting of the operational data. Operational database must be powerful enough to support thousands of concurrent transactions. It is optimized for a set of operations in short periods of time, such as addition, modification, deletion, and other operations. The client application affects the data in the database. Online transaction processing access atomic, detailed and current data in operational database.

Data warehouse database is specially designed to store integrated, historical, and consolidated data for business analysis. In general, a data warehouse system differs from an operational database in at least two major aspects. First, a data warehouse system is to serve online analytical processing in which complex, large, and ad-hoc queries occur frequently. Unlike the online transactional query that accesses only a small amount of data at a time, the analytical query needs to access many tuples per table, but users do not update the data. The design is not suitable for real-time operational system. The second important aspect of data warehouse is data integration. It is a huge data container for current and historical data provided by a number of existing data sources. After data is extracted from data sources, it needs to be cleaned, integrated, and aggregated before it is loaded into the data warehouse. Thus the data here is not the original data.

Due to the differences between operational database and data warehousing system, the hardware utilization is totally different. In Figure 2.5 we can see that the patterns of resource utilization are significantly different. As the result, it is impossible to combine both OLTP and OLAP in a server at the same time. Due to the reason, we have to implement the two functions in different servers.

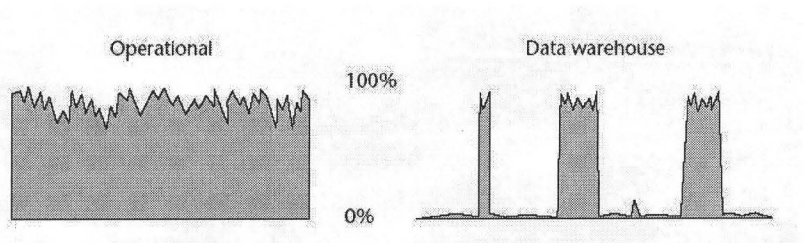


Figure 2.5: Hardware Utilization in Different Environments

We give a summary table for the comparison of operational database systems with data warehouse systems in Figure 2.6.

	Operational Database System	Data Warehouse System
Data	daily activities data dynamic data detail read & update real time update	historical data large static data aggregated read only bulk load
Design	transaction oriented relational normalized data model	subject oriented multidimensional or relational denormalized data model
Process	online transaction processing transaction driven simple query	online analytical processing analysis driven complex and ad-hoc query
User	clerk, operator	analyst, manager, executive
Size	MB to GB	GB to TB

Figure 2.6: Operational Database System vs. Data Warehouse System

2.5 Data Warehouse Design

In this section we briefly discuss various designs of data warehouses.

2.5.1 Virtual Data Warehouse

In a virtual data warehouse, the end-users utilize a network to access the organization's operational databases directly from client computers. The virtual data warehouse is still subject-oriented, however it has only fresh-valued and detailed data collections for support of the daily operations.

The advantages of a virtual data warehouse include:

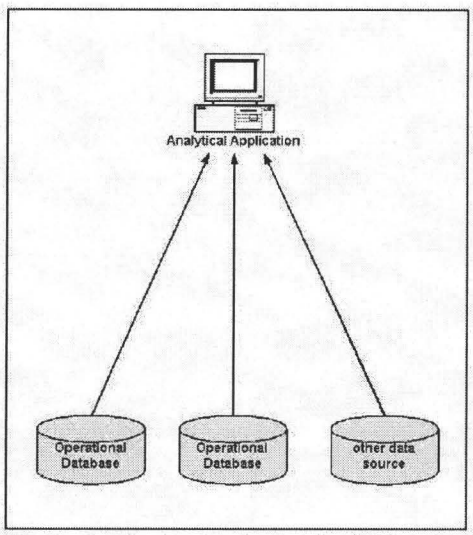


Figure 2.7: Virtual Data Warehouse

1. It is not necessary to develop additional tools for data extracting, cleansing, transferring, and loading.
2. End-users can access the most current corporate information.
3. There is no need to add extra hardware to hold another copy of analytical data.
4. There is no or less data redundancy since we use the operational database systems.
5. It is not necessary to add more human resources to maintain additional servers.

The disadvantages are:

1. The operational databases are designed for OLTP. It is not necessarily suitable for OLAP in many features. It may not meet all analytical requests.
2. There could be thousands of OLTP users accessing an operational database at the same time. If OLAP users share the same database with OLTP users, it

will affect OLTP's performance. Since most OLAP's analytical model usually takes a very long time and access many data records, end-users' access time is unpredictable.

3. Since there is no prior data cleansing and integration in advance, the data quality cannot be guaranteed. The data could be incomplete and/or inconsistent.
4. OLAP needs historical data for comparison and trend forecasting. However, operational databases do not keep the historical data, only the current data.
5. Analytical processing needs to access many records in database. Since the data comes from several different sources, the network influence is much heavier than in traditional database systems.

2.5.2 Centralized Data Warehouse

Centralized data warehouse is the most popular data warehouse architecture. It is depicted in Figure 2.8. A unique large database holds the integrated data loaded from different sources. The data is designed and modeled into a denormalized relations for data analysis.

Most organizations like to build a centralized data warehouse although many of their components (branches, offices etc.) may be geographically separated. A centralized data warehouse is a single physical repository that keeps all enterprise level data extracted from multiple operational systems and merged together in one place.

The advantages of a centralized data warehouse are:

1. People will use the same data for different analysis and decision making, because the data are unique. This enforces consistency.

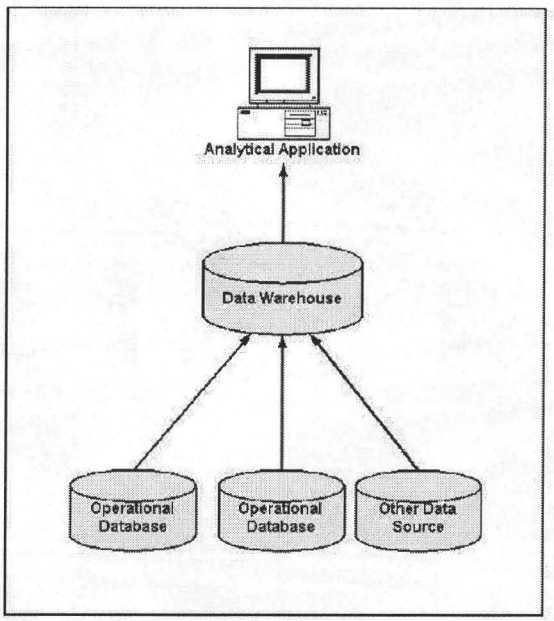


Figure 2.8: Centralized Data Warehouse

2. It separates the complex ad-hoc and time-consuming analysis queries from the operational database. If the OLAP queries were applied on the operational database directly, those queries could last a couple of hours and decrease the performance of operational database sharply. Thus the OLTP end-users would immediately feel the impact of such slow-down. The centralized data warehouse avoids such detrimental impact on the OLTP system by building a new data support system.
3. The centralized data warehouse has a higher standard in terms of security and control to the data access, and can also be managed more easily.

There are also a number of potential disadvantages:

1. Since centralized data warehouse keeps all historical data in a database, there is also a need to load new data from all data sources periodically. The data warehouse size will keep expanding over time. Unlike traditional operational database, given that a centralized data warehouse needs to keep historical data, its size could be at the terabyte level.
2. In a centralized data warehousing environment, expensive and high configuration hardware is required to guarantee the quality of complex analysis function and the swift response time. At the same time, sophisticated software systems are necessary to support a robust data warehouse.
3. Since all organization users can access one data warehouse, it might result in response delays. The network will suffer high volume data transfer if all end-users connect to the same destination at the same time. A fast and strong network is another important factor that has to be considered in the whole architecture. High performance networks are quite costly.
4. Since we only keep one unique data warehouse, the data could not be reached if the system goes down. Furthermore, the recovery time may be enormous. It is not very flexible because of its centralized nature.
5. Lastly, the initial build and load of data into a centralized data warehouse is very costly in time and people.

2.5.3 Distributed Data Warehouse

Most companies could adopt a centralized data warehouse, but in certain situations a distributed data warehouse would be more suitable and advantageous. Distributed

data warehouse architecture is depicted in Figure 2.9. First, distributed data warehouse may be better for an organization that spreads different business data over multiple locations for various types of business. For instance, if each branch has a unique business and specific requirements. Second, some companies would like to spread the whole data on different sites if they place heavy emphasis on data security and safety. However, we have to admit that only less than 5% of companies will eventually take the distributed approach to build data warehouse system.

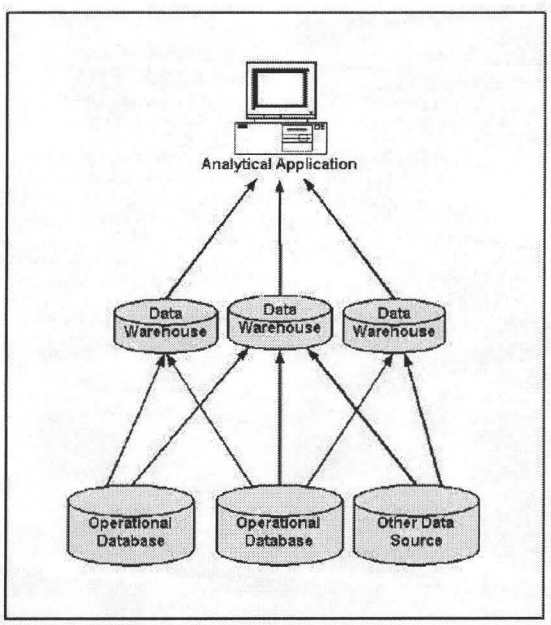


Figure 2.9: Distributed Data Warehouse

As we previously mentioned, the huge centralized data warehouse incurs more hardware and software costs because of its large setup and maintenance expenses, and the inflexible centralized characteristics. After some system implementation, most people have already noticed a centralized data warehouse size tends to be extremely

large and expands fast. From a technical view, for the data warehouse to retain the good performance, the company has to invest more funds and resources. The maintenance job also becomes difficult for such a large system. For instance, DBAs need much longer time to backup a big data warehouse.

To improve performance and reduce the burden on a single resource, the distributed database system allocate the data into several small databases. By the same token, a distributed data warehouse can be set up. In this way, a couple of individual data warehouses work together to provide a global view of the enterprise information to users. Both distributed database and data warehouse are more complex and difficult to manage. Because of data load balance problem, the performance and quality of most distributed queries are usually poor.

The advantages of a distributed data warehouse are:

1. Once the data warehouse expands, we always can add one more site without affecting the current system and analytical operation.
2. Since the distributed data are stored at several sites, some data are still available even if one site is down.

Here are the disadvantages of a distributed data warehouse:

1. Some query may need to use data across several servers to finish. With data being transferred from one site to another over the network frequently, the data security is pretty weak.
2. The management of several remote data sites is very challenging. The data at all sites may not be refreshed or updated at the same time which causes serious problems of data balance among all distributed sites.

3. More human resources would be needed to maintain and support every site.
4. The whole data warehouse system performance tends to be poor. The large data flow might also increase the burden on network.

2.6 Data Marts

Data mart is a subset of data warehouse designed to help certain department business or strategic decision system. Similar to data warehouse, data mart is also a subject-oriented, time-variant, integrated, and summarized repository of data from other data sources to support strategic decision making. Unlike data warehouse, data mart is designed to serve a particular group of people in terms of decision making and future planning. Depending on the data acquisition, there are two different architectures. The data could be directly derived from operational database or from a enterprise level data warehouse.

The table in Figure 2.10 summarizes the different aspects of data mart in comparison to data warehouse:

	Data Warehouse	**Data Mart**
Function	corporate functions cover all functions	department functions address specific requirement
Data	huge historical data amount summarized level all corporate subjects	modest historical data amount refine level some specific subjects
Structure	corporate data structure	department data structure
Design	large cost and long time	low cost and short time
Performance	slow response	improved performance

Figure 2.10: Differences between Data Warehouse and Data Mart

The following benefits of data mart can help resolve some problems that data warehouse cannot overcome.

1. Data warehouse covers the all business functions while data mart only focuses on a specific function, subject, or category. It brings flexibility to individual divisions. People can decide their own data schema and the time to refresh data mart without affecting other groups.
2. Data in a data mart are normally at a refined level for business usage.
3. Unlike a huge data warehouse, data mart size is much smaller. It can thus reduce the initial implementation cost, effort, and time. At the same time, it improves the response time and cuts maintenance costs.
4. Data mart separates transactions to different destinations, reducing the bottleneck of accessing one data warehouse simultaneously.

Basically, there are two data mart architectural models as described in the following subsections.

2.6.1 Enterprise Data Marts Architecture

With an enterprise data marts architecture, data are extracted, cleaned, and loaded from data sources into data marts directly. The data in a data mart are of interests only to a particular group of knowledge professionals. See Figure 2.11.

For instance, individual team or division could create their own data mart that meets their own specific data requirements. Compared to the whole organization requirement, such a request should be small. The development time could be much shorter since we do not collect all data for every department. Some data marts could be put into production flexibly even before other marts are in development. Since data

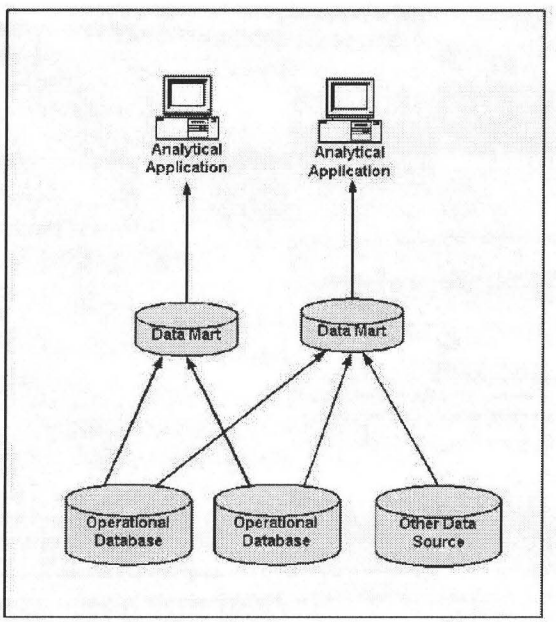


Figure 2.11: Enterprise Data Marts Architecture

marts only serve a specific group, it provides more detailed, relevant, and accurate data for the decision makers. Furthermore, the team would have full control ability over their own departmental data. Since data mart is a local system not in conflict with other departments, the people can decide when they need to refresh the data. Certainly it will improve the availability and performance as well.

2.6.2 Data Warehouse and Data Marts Architecture

The second architecture depicted in Figure 2.12 is a combination of data warehouse and data marts. The data marts are based on the existing data warehouse. The data from operational system and all other external informational systems are loaded into the data warehouse first after cleansing. According to specific business request, data

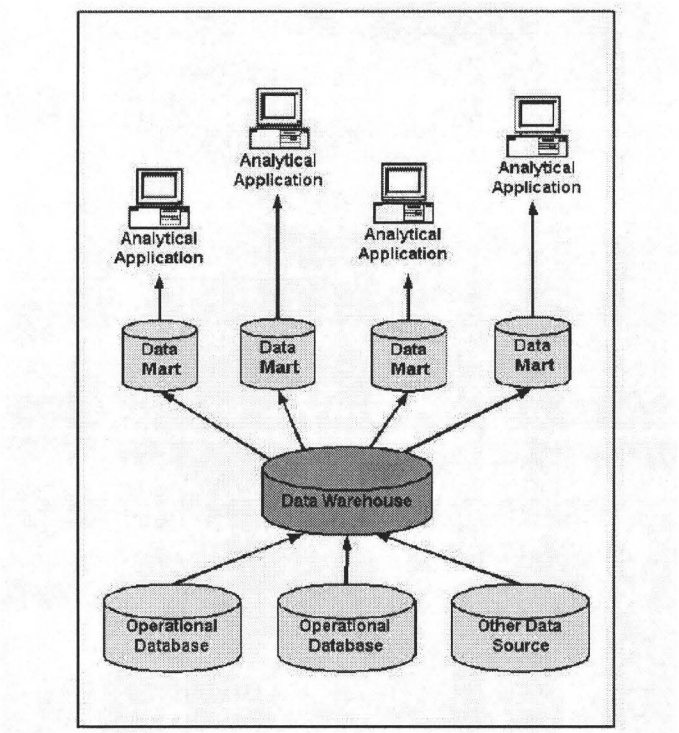


Figure 2.12: Data Warehouse and Data Marts Architecture

mart then retrieves customized data from data warehouse. Obviously, the data are integrated and summarized since they had been cleaned before it is loaded into the data warehouse. This model combines enterprise and department information in one server. If an individual division needs information related to the whole organization, they can acquire it from the data warehouse as well.

There are some advantages in data warehouse and data mart architecture:

1. Data warehouse stage already finishes the data integration and cleansing. Division users do not need to repeat the complex ETL jobs when they implement their department data mart.

2. It is easy to add or drop a data mart without any impact on the data warehouse system.
3. To recover a data mart is possible since data warehouse keeps all original data.

Of course, this architecture also comes with its own disadvantages:

1. There is a data redundancy issue between data warehouse and data mart.
2. It is complex to maintain data sync between data warehouse and data marts.
3. The company may also need more resources to manage the increasing numbers of data marts.

2.7 Metadata Management

In a traditional database system, metadata means the data about data. A metadata contains data models, element data definitions, and database description.

Data warehouse system is a complex system, consisting of various data from different sources. The data will be accessed by different groups of users and they do not have complete knowledge about the original business and data. Metadata management is an important activity necessary for understanding and maintaining of such a complex system. Actually, metadata are stored in several reference tables in the data warehousing system.

Similarly to a card catalog, metadata helps people navigate the warehouse and search for information. A metadata management is very important for a data warehouse that keeps 5-10 years worth of data. The data schema could changed a lot during such a long period; the metadata will record all the changes and history.

Since data warehousing system includes several data areas, we need to build at least three metadata repositories:

- source system metadata,
- staging data area metadata, and
- data warehouse database metadata.

In a data warehouse environment, metadata are classified into operational type and business type.

1. Operational metadata contain information about the construct, process or operation of the warehouse, such as data structure, table and index definition, data refresh, security, and audit message. Operational metadata are very useful to system design, architecture establishment, project development, and routine maintenance in the future.
2. Business metadata should contain all business information, such as data source, data ownership, business definitions and business terms. It is a useful information for end-users. Business metadata can help navigate information about business description.

A metadata repository should be set to maintain all metadata information related to a data warehouse system. It is very important to keep a clear historical metadata information since the data in a data warehouse system come from various data sources. At same time, data warehouse projects usually last a couple of years. The data source could keep changing during the period. End-users could be easily confused by a large volume of future data. Metadata are critical to system design, project administration, and routine maintenance.

Chapter 3

Data Model of Data Warehouse

3.1 Definition

A data model is a collection of conceptual tools. It describes data schema, data relationships, data semantics, consistency constraints, and data itself as well [9]. Data model serves as a bridge between business requirement and physical database implementation. With a good data model, end-users could completely understand the data structure and the associated business of the organization. Data model is very important and useful for developers to build an efficient data warehousing system.

3.1.1 Data Model Types

Data models can be classified into three categories: conceptual, logical, and physical (see table in Figure 3.1).

Model	Description	Entity	Perspective
Conceptual	Semantic Model	Business Entity	User
Logical	Logical Data Model	Data Entity	Modeler
Physical	Data Implementation	Table/Column	DBA

Figure 3.1: Data Model

1. **Conceptual Model**

   Conceptual model is also called domain model by some authors. It represents a high level data model approach. The purpose of a conceptual model is to identify and describe the business concepts at the beginning of a project.

   Conceptual data model includes only the important entities and relationships among them. Usually the data attributes and primary and foreign keys will not be specified within this model.

2. **Logical Model**

   Logical model includes both graphical depictions and textual definitions. The main function of logical model is to transform business requirement into a structured representation of data.

   Logical data model not only includes all entities, but also relationships among them. Both primary and foreign keys need to be specified in this model. Data modelers describe data in a detailed level in this stage, including the normalization and many-to-many relationships, but not the physical implementation. The conceptual data model and the logical data model are combined into a single step in a data warehousing system modeling.

   The major steps in design of the logical data model are the same as in the database design: identify all entities and primary keys, identify all relationships between entities, identify the attributes of the entities, resolve many-to-many relationships and normalize the model.

3. **Physical Model**

   Physical model represents the details of data implementation, including database

tables, indices, relationships, rules, and security considerations. We use the physical data model to deploy database systems. It may also include graphical depictions and textual definitions.

Within this model, the data modeler will describe how the logical data model will be implemented in the detailed database schema.

To design a physical data model, the entities are converted to tables and attributes to columns. The relationships are converted to foreign keys. The physical data model needs to be modified according to physical constraints.

3.1.2 Objectives of Data Model

The main goal of data modeling is to design a more efficient and effective database system. Some important objectives of data model:

1. builds a consistent overview of the business rules and system requirements,
2. helps understand and evaluate the business requirements and provide feedback,
3. identifies the dependencies among the data,
4. identifies the unaddressed attributes and relationships,
5. reduces the potential risk.

3.2 The Types of Basic Data Models

In this section, we briefly discuss the major types of basic data models and how the relate to data warehousing.

3.2.1 Entity-Relationship Data Model

It is customary in the database literature to abbreviate the term Entity-Relationship as E-R.

"The E-R data model is a way to create relational database views and is developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database. E-R model includes information about the relationships among these objects." [9]

However, E-R model is not suitable for data warehousing. Since end-users of data warehouse are not operational system users and do not have original business knowledge, they could not understand an E-R model which is related to the organization's business. E-R modeling is designed for day to day operations in an organization, not optimized for complex, ad-hoc queries which are used by data analytical functions. Most importantly, for a large data retrieval system, using E-R model can lead to deteriorating performance.

3.2.2 Relational Data Model

"The relational approach to database design begins with a collection of tables representing data and relationship among those data. Relational technology uses keys and foreign keys to establish relationships between rows of data. The value of the relational model for database design for the data warehouse is that there is discipline in the way the database design has been built, clarity of the meaning, and use of the detailed level of normalized data." [9] How relational data model relates to data warehousing is discussed in the section **Data Model for Data Warehouse**.

3.2.3 Dimensional Model

"A dimensional model contains the same information as an E/R model but packages the data in a symmetric format whose design goals are users understandability, query performance and resilience to change. Dimensional model is a logical design technique that seeks to present the data in a standard framework that is intuitive and allows for high performance access." [10] How dimensional data model relates to data warehousing is discussed in the section **Data Model for Data Warehouse**.

3.2.4 Relational Model vs. Dimensional Model

We summarize the main differences between relational model and dimensional model in the table given in Figure 3.2.

Relational Model	**Dimensional Model**
business rules	reflect enterprise view
strategic requirement	functional needs
optimized for flexibility	optimized for legibility
data maintenance	data visualization
structure from logical data model	aggregation organization structure
fairly normalized	denormalized
detail in content	business dimension

Figure 3.2: Relational Modeling vs. Dimensional Modeling

3.2.5 Data Model for Data Warehouse

The purpose of a data warehousing system is to generate user-oriented reports. The main purpose of data model is to build an efficient access to data. Therefore, the data model for a data warehousing system is relatively simple. However, careful data source analysis is essential to create a good data model which helps to retrieve

the most useful and easily understandable data for analysis. In a data warehouse system, data modeling is a critical factor, which directly impacts the data warehouse development, future maintenance, and system quality.

There are two widely used data models for data warehouse database design: relational model and dimensional model. Most people consider relational model as "Inmon" approach while dimensional model as "Kimball" approach to data warehousing. For a long time discussions of merits of each and their comparisons have been going on. Both approaches have their own strengths and weakness.

Relational data model is hard to understand from data warehouse end-user standpoint, but it is good for a long term approach in building an enterprise data warehouse. On the other hand, although dimensional data model looks too sparse, it is the best approach for a short term data warehouse. However, we will not focus on which data model is superior in this thesis. In fact, the choice of data model really depends on the particular characteristics of the data warehousing system to be designed.

In the case study presented later, we will use dimensional data model as the design tool. The main purpose of dimensional modeling is to present the data in a standard framework by a logical design technique and help visualize the data. To help conduct easy and flexible business analysis, most dimensional models are specifically designed for a high-performance access by slicing data along certain dimensions according to the aim of the performed analysis. Also, it creates a simple and easy means to understand and navigate the complex data for the end-users.

3.3 Dimensional Model in Data Warehouse

Dimensional database is used to analyze data along certain dimensions.

Dimensional model typically consists of a fact table and many dimension tables. Since a single fact table is surrounded by numerous dimension tables, it is often called *star schema* (see Figure 3.3).

3.3.1 Building a Dimensional Model

According to Kimball, there are four basic steps to build a successful dimensional model.

1. Identify the business process
2. Identify the level of detail needed (grain)
3. Identify the dimensions
4. Identify the facts

3.3.2 Fact Table

Fact is a numeric and additive datum that can be analyzed.

Typical data usually

1. are about an observation in the marketplace; or
2. serve as numeric measurements of interest; or
3. represent facts associated with natural business dimensions; or
4. quantify data described by dimension tables; or
5. contain keys and measures (two types of columns) for data tables; or
6. contain either detailed level facts or facts that have been aggregated.

A fact table is an essential table in each dimensional model and consists of business measurements. The primary key of a fact table is usually composite. The composite key is made up of all the foreign keys of the surrounding dimension tables.

3.3.3 Dimension Table

Dimension describes an enterprise business attributes represented as hierarchical and categorical information. Three most common dimensions are time, geography, and product.

Characteristics of dimensional table include:

1. Dimensions consist of levels, which define the hierarchy of a dimension.
2. Constraints are used in forming fact table.
3. Dimensions start with the most detailed aggregation level if necessary.

Dimension tables are sometimes called lookup tables. They represent analysis views.

3.3.4 Star Schema

The star schema (see Figure 3.3) is named for its resemblance to a star, with one large central table (fact table) and a set of smaller surrounding tables (dimension tables) radiating from it. The fact table is a conjoint that consists of dimensional key and measurement of data. Key and measurement are two kinds of columns in fact table. There is a couple of dimensional tables that represent one major categorical information of the fact table. The measurements are captured into fact table and attributes of those measurements are stored in dimension tables. The fact table represents some business measurements while the dimension tables represent who, what, where, when and how of an occurrences.

The advantages of star schema are:

1. It presents a direct and clear mapping between the schema design and the business entities which will be analyzed. It is an easily understandable format.

2. Since star schema is easy to change and extend, it simplifies the development process.
3. Most business intelligence tools in the market support star schema.
4. Basically, it helps secure a swift response time and highly optimized performance.

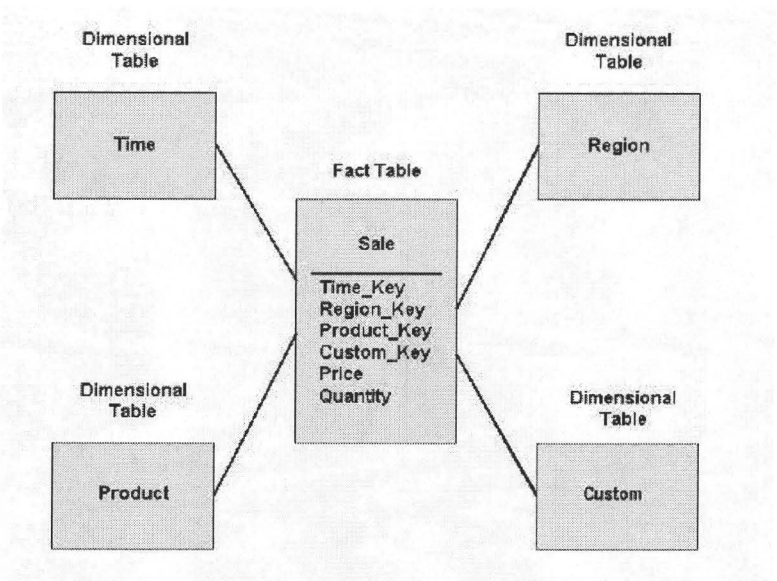


Figure 3.3: Star Schema

3.3.5 Snowflake Schema

Another schema that is widely used in data warehousing system is snowflake schema (see Figure 3.2). The schema diagram resembles a snowflake, so it is called snowflake schema. In general, the snowflake schema is also a kind of star schema, but a little more complex one — its dimension tables are built in normalized structure. The

main purpose of normalized structure is to eliminate data redundancy in dimensional tables. Therefore, a large dimension table may be partitioned into several dimension tables in snowflake schema. This leads to a proliferation of foreign keys increasing the complexity of the queries. For a large data warehousing system, I think we need to abstain from using snowflake schema. Although it may save some storage space, the final analytical process will take longer time due to the connection among fact table, dimension tables, and sub-dimension tables.

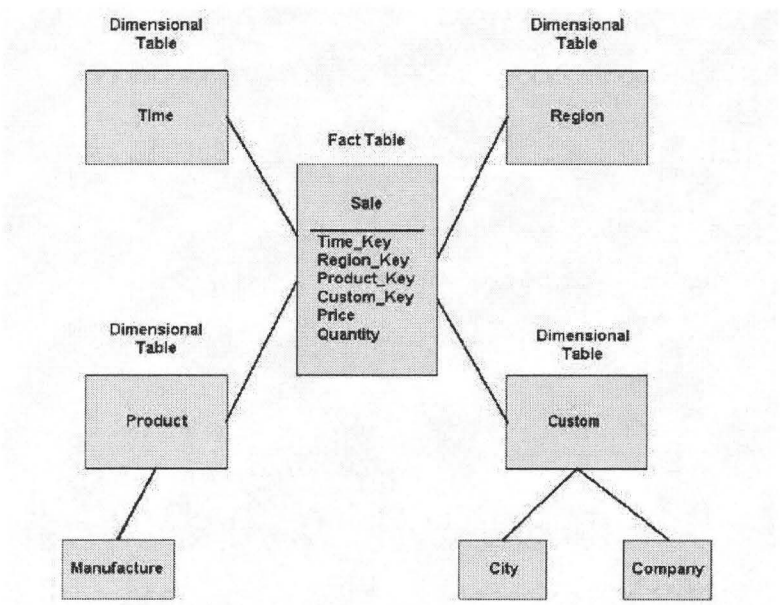


Figure 3.4: Snowflake Schema

The advantages and disadvantage of snowflake schema in summary:

1. It eliminates redundancy by normalizing dimension structure.
2. It reduces some storage cost, but at the expense of lower data browse performance.

3. More foreign keys are required due to partitioning of dimension tables not in a normal form.
4. It increases the complexity of query and query effectiveness.

3.3.6 Granularity

One of the most important data warehouse design aspects is determining the appropriate level of granularity. Granularity is the relative depth, scale, or size of an object being characterized. In data warehouse, granularity refers to the level of summary or detailed data held. The more detailed data is required in data warehouse, the lower the granularity level needs to be designed. Otherwise, the lese detailed the data is, the higher the level of granularity.

The data warehouse data size will be affected sharply by the granularity level. We can answer most kinds of detail queries if we store data by low granularity, but it needs a lot of space. On the other hand, if we use high granularity policy, the resulting system will not able to answer some detail queries. However, it will save space and will be a more efficient means to represent data.

The fundamental issue of granularity is to set an appropriate level. The level of granularity should not be too high or too low and needs to cover most queries. To define an effective granularity, we need to determine which dimensions should be included and along where the hierarchy of the dimension information needed to be stored in the data warehouse.

3.3.7 Normalization vs. Denormalization

The issue of intentional denormalization of database for performance gains also affects data warehouse design.

Normalization

The process of normalization is to reduce redundancy, eliminate composite keys for partial dependency, and guarantee data consistency. Normalized data are constrained by the primary key/foreign key relationships to achieve a high degree of data integrity. Typically, operational databases are highly normalized.

The benefits of the normalization include:

1. Data is integrity since there is no redundant data.
2. Data update is faster because there are fewer indexes per table.
3. Because of the fewer columns in the tables, we are able to maintain faster index and sorting.
4. Normalized tables produce rapid and efficient joins which help to optimize queries (of course, with denormalization we might not need so many joins).
5. Because table lock affect less data, it improves concurrency resolution.

Denormalization

Denormalization is the process to re-introduce the redundancy in order to improve the performance. As the data for query purpose is typically stored in a single row, the retrieval is much faster.

Due to the large amount of data involved, data warehouses often use denormalized (star schema) or partially denormalized schemas (snowflake schema) to design data warehouse database. Since most data is read-only, the most important fact of design is to optimize query performance. In contrast to traditional database design strategy, the main idea is to break the rules of normalization to increase performance and simplify ad-hoc query in data warehouse system. Denormalization actually saves the

redundant data to make a much faster query. It is acceptable to lose track of the relationships among atomic data during the denormalization process.

The most useful techniques of denormalization include:

1. Duplicating data;
2. Providing summary data;
3. Splitting tables into horizontal or vertical partitions;
4. Creating denormalized views to simplify reporting.

Chapter 4

Online Analytical Processing

4.1 OLAP definition

E.F. Codd introduced the term **On-Line Analytical Processing** (or OLAP for short) in 1993, as a special category of data processing focusing on intuitive and multidimensional analysis of summarized information for decision support system [11]. OLAP is an analytical tool for querying large database, like data warehouse. OLAP allows users to get business intelligence and information from data warehouse systems by query and analytical applications.

The information and statistics are not directly visible inside the simple data directly retrieved from a data warehouse. OLAP is a highly dynamic and interactive method to derive or reveal business trends based on the data from a data warehouse. It is dynamic and adaptable, supporting various analytical activities. OLAP is to provide business information, such as "what was the product sales in Toronto in the last 2 years?" , "what is the difference between Toronto and Hamilton in product sale?" and "which group of customers will be the most profitable for the company in the future?". OLAP navigates through historical data and extracts useful patterns, then forecasts the trend.

When E.F. Codd gave the definition of OLAP, he also listed 12 basic characteristics of OLAP [11]:

1. Multidimensional
2. Transparency of the server
3. Accessibility
4. Stable access and performance
5. Client-server architecture
6. Generic dimensionality
7. Management of data sparsity
8. Multi-user
9. Operation on dimension
10. Intuitive manipulation of data
11. Flexible posting and editing
12. Multiple dimensions and levels.

Among these characteristics, we must pay attention to the new features around the concept of multidimensionality. It is the most important aspect of this new technique. Multidimensional design is flexible and easy to analyze. One data dimension represents a feature of business analytical data. Typically, there are about 10 various dimensions from which data could be analyzed in a real world. Because the conceptual view of OLAP models is multidimensional, we align model design and analysis around this characteristic. As a result, end users can easily manipulate such multidimensional data models. By changing a couple of dimensions in an analysis module, they can get different business results. It is simple and frequently requested

in a decision support system. Moreover, people can add or drop a dimension without affecting other functions.

In 1997, OLAP was further defined in details by *OLAP Council*: On-Line Analytical Processing (OLAP) is a category of software technology that enables analysts, managers and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information that has been transformed from raw data to reflect the real dimensionality of the enterprise as understood by the users [16].

Online analytical processing is used to provide planning for optimizing the business by a series of queries. OLAP queries typically are complex calculations, such as feature comparisons over time and business forecasting. OLAP usually analyzes data using multiple dimensions, like the view of product, region and time. This kind of data is called multidimensional data in data warehousing system.

4.2 Background and History

Let us review some important historical moments related to OLAP.

1970 — E.F. Codd proposed relational model.

1980 — SQL became a commercial success.

1993 — E.F. Codd coined the term OLAP.

1993 — Excel started to offer pivot tables.

1997 — MOLAP vs. ROLAP debate.

1999 — SQL-99 started to offer some OLAP functionality.

In the following, we will discuss these notions.

4.3 OLAP vs. OLTP

OLTP is specifically designed to capture daily business information of an organization through recording basic business process data. Second, OLTP is optimized for short transaction performance, and focuses on update performance and maximizing concurrency. The schema design is typically normalized with many relations.

OLAP is to support business analysis by providing a multidimensional view of data. Unlike OLTP, OLAP queries are large, complex and ad hoc. They will not modify any record, but queries retrieve hundreds of records. In an analytical system, query performance is the primary priority. OLAP database reintroduces data redundancy while sacrifices space to speed up queries. The basic approach is to denormalize data in database.

In the table in Figure 4.1, we illustrate some important contrasting features between OLTP and OLAP.

OLTP	**OLAP**
normalized model	denormalized model
query access few records	query access numerous records
read and write query	mostly read-only query
detailed data	summarized data
current data	historical data
high degree of concurrency	low degree of concurrency
comparatively small database	huge database
optimized for business	optimized for analysis
staff, operator	analyst, manager

Figure 4.1: OLTP vs. OLAP

4.4 Cube

Unlike relational databases that use two-dimensional data structures, a data cube consists of more than two dimensions which represent various perspectives of data, like region, time, customer, etc. A cube is typically organized and structured in a hierarchical and multidimensional arrangement. OLAP cubes can have numerous dimensions and levels of data. Usually the data is pre-summarized into various consolidations and subtotals (aggregations) for the best query performance. It is an easy and fast means for users to visualize the multidimensionality of OLAP information data. The data cube cells represent a couple of analytical object measures. There are different cubes for various types of business in an organization.

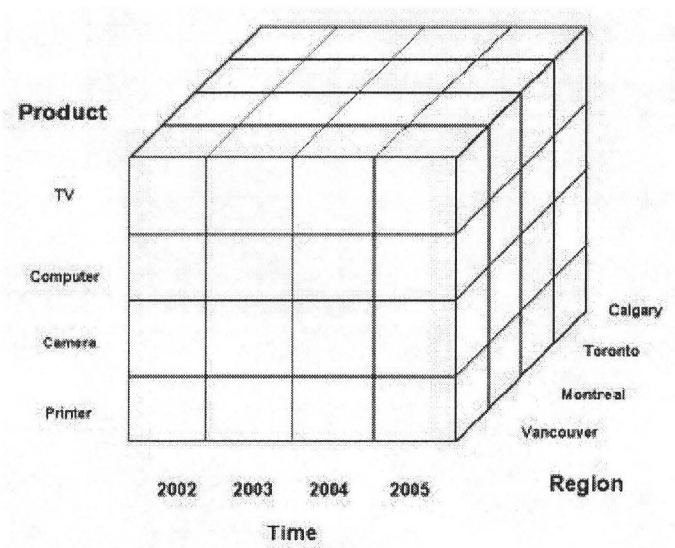


Figure 4.2: Cube

In Figure 4.2 a simple three-dimensional cube consisting of time, region and product dimensions is shown. There are several import procedures for manipulating the data

cubes. In the following we discuss the most important ones.

4.4.1 Pivoting

Pivoting is a procedure that interchanges certain dimensions of the cube. Consequently, the data is represented by a different multidimensional view.

(A) Let us first discuss a two-dimensional rotation example depicted in Figure 4.3.

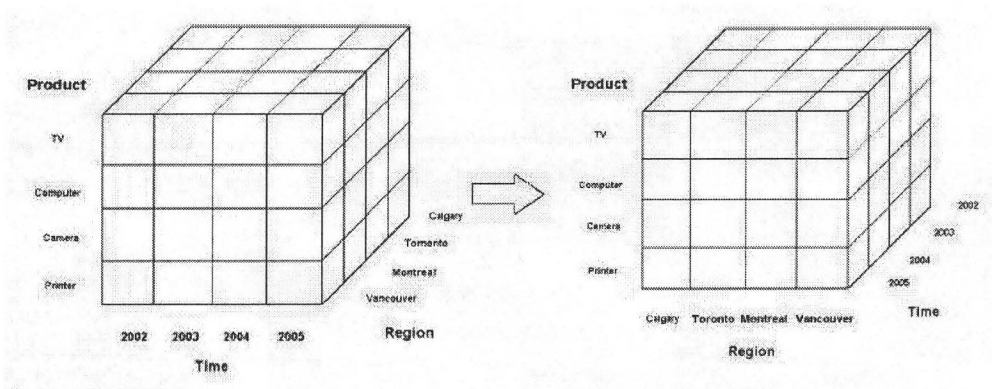


Figure 4.3: Cube Pivot

The table in Figure 4.4 represents the original view of the cube before pivoting.

	2002	2003	2004	2005
Calgary	30	50	60	70
Toronto	100	200	230	300
Montreal	50	230	330	300
Vancouver	70	30	60	90

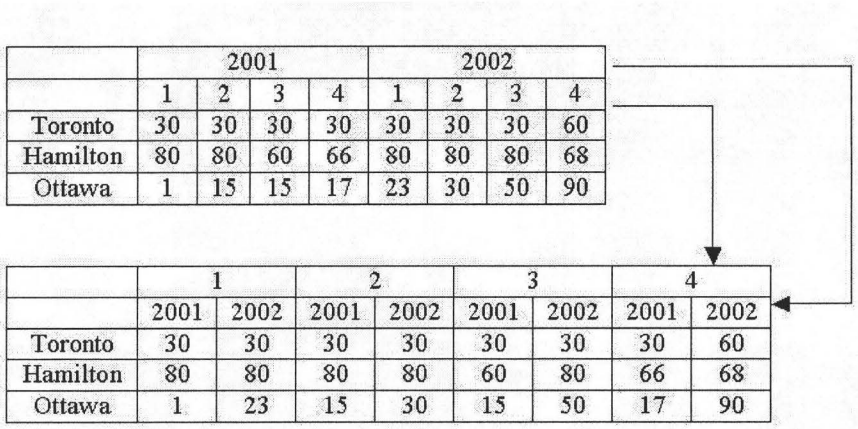
Figure 4.4: Original View for product TV

After pivoting, the report view is changed due to the new dimensional view presented in Figure 4.5.

	Calgary	Toronto	Montreal	Vancouver
2002	30	100	50	70
2003	50	200	230	30
2004	60	230	330	60
2005	70	300	300	90

Figure 4.5: Changed View for TV after pivoting

(B) The second example illustrates two different granularity levels rotating in the same time dimension. After the rotation, end-users can compare the analytical data of the same quarters among different years, see Figure 4.6.



	2001				2002			
	1	2	3	4	1	2	3	4
Toronto	30	30	30	30	30	30	30	60
Hamilton	80	80	60	66	80	80	80	68
Ottawa	1	15	15	17	23	30	50	90

	1		2		3		4	
	2001	2002	2001	2002	2001	2002	2001	2002
Toronto	30	30	30	30	30	30	30	60
Hamilton	80	80	80	80	60	80	66	68
Ottawa	1	23	15	30	15	50	17	90

Figure 4.6: Granularity Levels Rotation

4.4.2 Roll Up & Drill Down

Roll up acts as an aggregation across a consolidation dimension, navigating data from detail to aggregated. It works very much like the "group by dimensions" clause in the SQL language.

Drill down is the reverse procedure of roll up, from aggregated to detail view of data along a certain path.

The diagram in Figure 4.7 demonstrates the respective actions of "roll up" and "drill down".

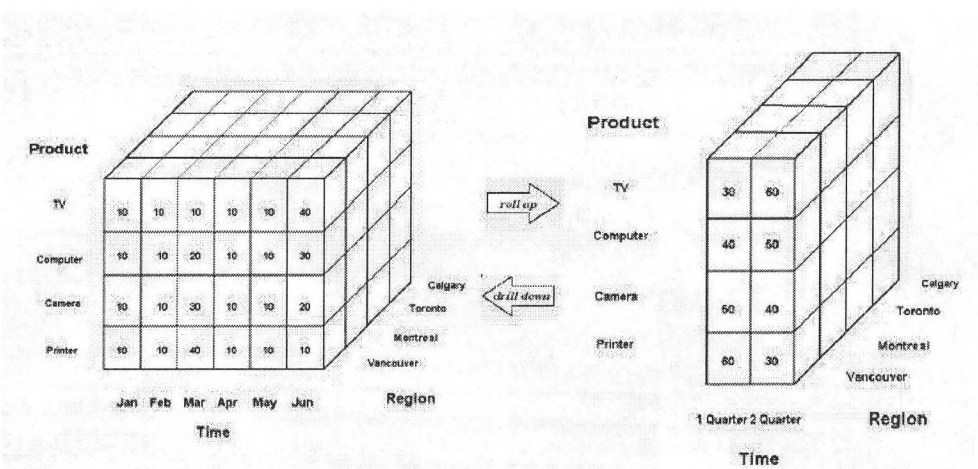


Figure 4.7: Roll Up & Drill Down

4.4.3 Slice & Dice

"Slice & Dice" is a means to reduce the dimensionality of complicated data, performing operations such as taking a projection of data. In this case it only takes a subset of dimensional data for an analytical purpose.

The example in Figure 4.8 shows the value of TV product of Vancouver in 2002.

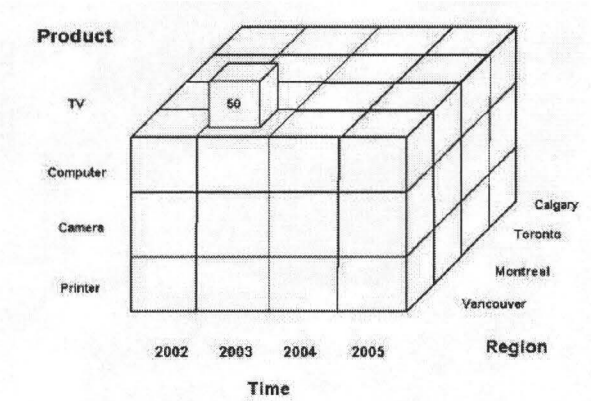


Figure 4.8: Slice & Dice

4.5 OLAP Technology

There are two most frequently used types of OLAP: **relational OLAP** (abbreviated ROLAP) and **multidimensional OLAP** (abbreviated MOLAP). Another technology referred to as **hybrid OLAP** (abbreviated HOLAP) appeared just recently; it attempts to combine the advantages of both MOLAP and ROLAP.

The different OLAP approaches lead to a number of variations in data structure, organization, and storage. ROLAP uses relational database (abbreviated RDB) while MOLAP uses multidimensional database (abbreviated MDDB).

Both ROLAP and MOLAP provide the end-users with a multidimensional view of the data. MOLAP server is capable of storing data in multidimensional storage and viewing the data multidimensionally. ROLAP server is able to transform request dynamically and access relational database by join/group algorithms, then return a multidimensional view to the end-user. However, there are no physical cubes existing on ROLAP server, only virtual ones.

A dimension is actually a point of view by people. However, the view is not chosen randomly. We select appropriate dimensions to serve specific analytic purposes. For instance, in order to analyze sales information of a product of an international company, we can select the time dimension and the region dimension. The measurement is the sales amount. In the Figure 4.9, a two-dimensional data organization in MDDBMS and RDBMS is shown. The time dimension has two levels: year and quarter.

Data in RDB

Region	Year	Quarter	Sale
Toronto	2001	1	30
Toronto	2001	2	30
Toronto	2001	3	30
Toronto	2001	4	30
Hamilton	2001	1	80
Hamilton	2001	2	80
Hamilton	2001	3	60
...	...	...	...
Toronto	2002	1	30
...	...	...	...
Ottawa	2002	1	15

Data in MDDB

	2001				2002			
	1	2	3	4	1	2	3	4
Toronto	30	30	30	30	30	30	30	60
Hamilton	80	80	60	66	80	80	80	68
Ottawa	1	15	15	17	23	30	50	90

Figure 4.9: RDB & MDDB

This diagram illustrates the differences between a ROLAP server and a MOLAP server. ROLAP data is in the form of columns and rows in a spreadsheet. MOLAP cube is a logical and multidimensional model which may contain numerous dimensions and levels of data.

4.5.1 ROLAP

ROLAP methodology relies on storing and manipulating data in relational databases that are known for their ability to scale up to a large amount of data. Each analytical

operation of slice or dice works like a "where" clause in the SQL language. It is called relational OLAP because it uses a relational database. The submitted SQL queries return the results to the end-user in a multidimensional view.

ROLAP approach is capable of dealing with a large amount of data easily. It can take advantage of inherent functionalities of the relational database system. However, the performance of ROLAP is relatively slow since each ROLAP query is generated by one or multiple SQL queries.

ROLAP stores facts as records by relational technique which is a quite mature technology after years of refinement. It is also suitable for sparse data. If there is no fact, there is no storage required. However, it may needs more indices to improve the speed of queries.

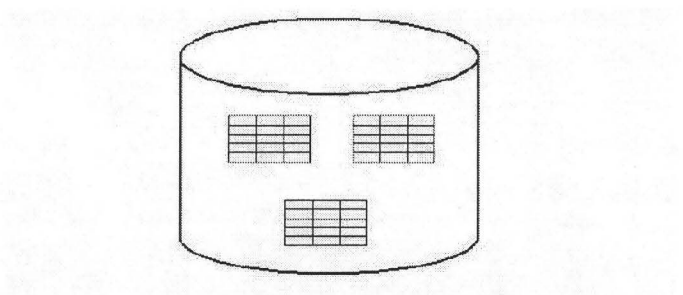
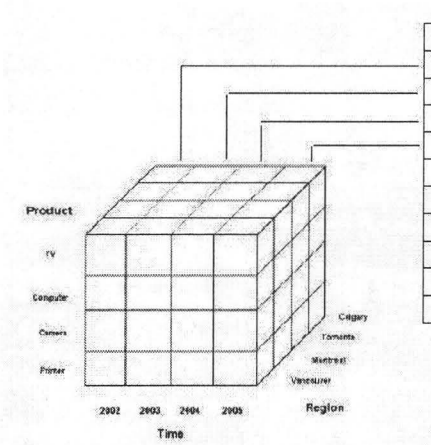


Figure 4.10: Relational Database storing multidimensional information

In order to describe multidimensional data by relational tables, ROLAP separates multidimensional structure, using dimension tables and a fact table to represent the multidimensional information, see Figure 4.10.

Fact Table

A Fact table captures the fact measurement when several dimensions are intersected at a certain point. It is illustrated in Figure 4.11, where the dimensions are Product Key, Region Key, Time Key, and Sales. The measurement is the sales amount.



Product Key	Region Key	Time Key	Sale
1	1	1	100
1	1	2	200
1	1	3	300
1	1	4	400
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...
...	...	...	...

Figure 4.11: Fact Table

Except "Sales", all other attributes in the fact table are primary keys from the dimension tables. These attributes cannot be null in the fact table. This structure minimizes the storage of primary keys connecting to the dimension tables. The benefit is significant when most of the dimension tables contain many levels of granularity and large amount of records.

Dimension Table

Actually, a dimension table is a table to record the dimension information, mapping the dimensional design onto the relational model. Let us demonstrate the process of data mapping by the diagram in Figure 4.12. There, the data cube consists of the

product dimension, the time dimension, and the region dimension. We convert these dimensions to several dimension tables with the associated field names and possible data values: (1) Product Dimension Table (Product Key, Product Name); Values - TV, Computer, Camera and Printer (2) Time Dimension Table (Time Key, Time Name); Values - 2002,2003,2004 and 2005 (3) Region Dimension Table (Region Key, Region Name); Values - Calgary, Toronto, Montreal and Vancouver.

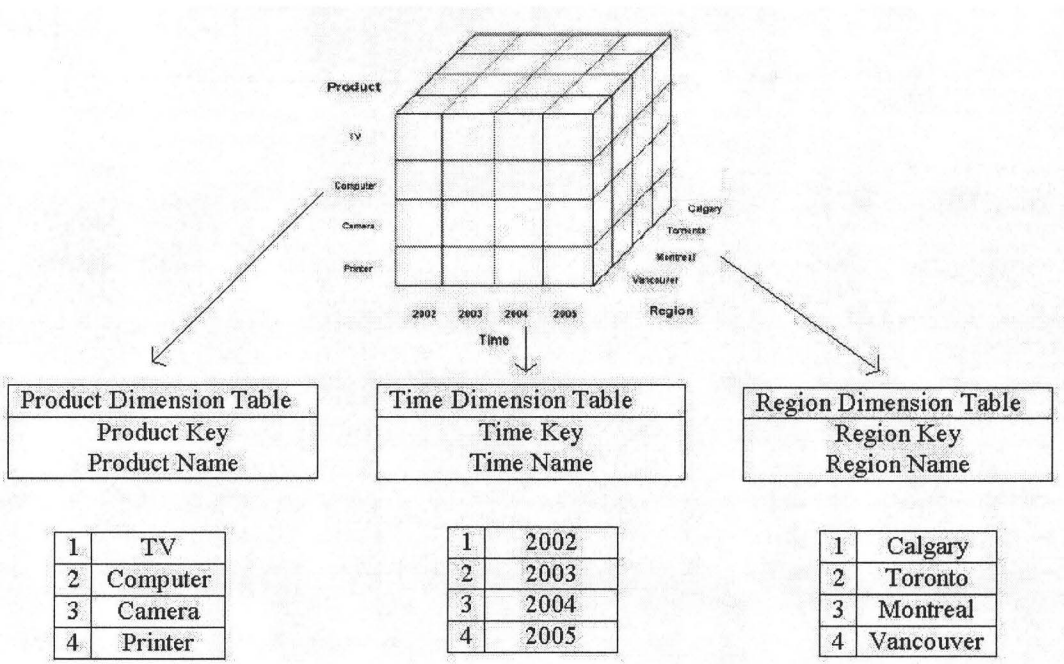


Figure 4.12: Dimension Table

If we connect the fact table and the dimension tables together, we can get the star schema. As demonstrated by Figure 4.13, it is a typical relational database structure.

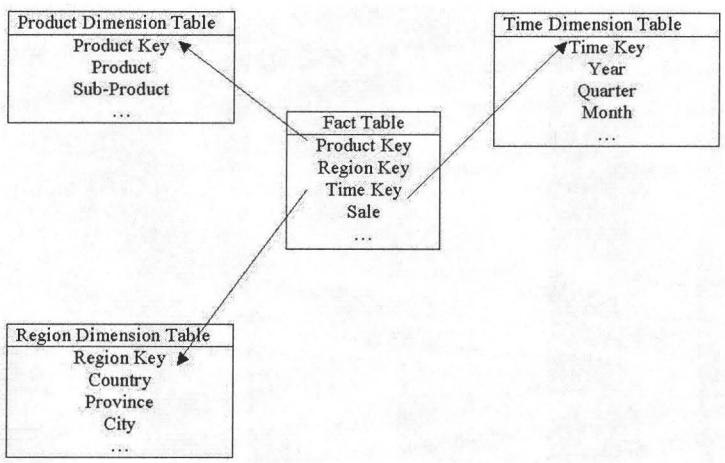


Figure 4.13: Generate Star Schema by Fact Table and Dimension Tables

ROLAP Space

ROLAP method has the advantage of lowering the storage requirement. Let us use the example in Figure 4.14 to demonstrate it. For the purpose of our illustration, we are assuming that there are 30 tuples in each of the dimension tables and each column length is 10 bytes. We can compute the space needed for the configuration of tables in order to store the data.

Fact table: 30 * 30 * 30 * 4 * 10 = 1080000 bytes

(*30 * 30 * 30 records, each record consists of 4 keys, each key has size 10 bytes*)

Dimension tables: (3 + 4 + 4) * 10 * 30 = 3300 bytes

(*2 tables have records with 4 columns of size 10, one table has records with 3 columns of size 10, and each table has 30 records*)

Total space required: 1080000 + 3300 = 1083300 bytes

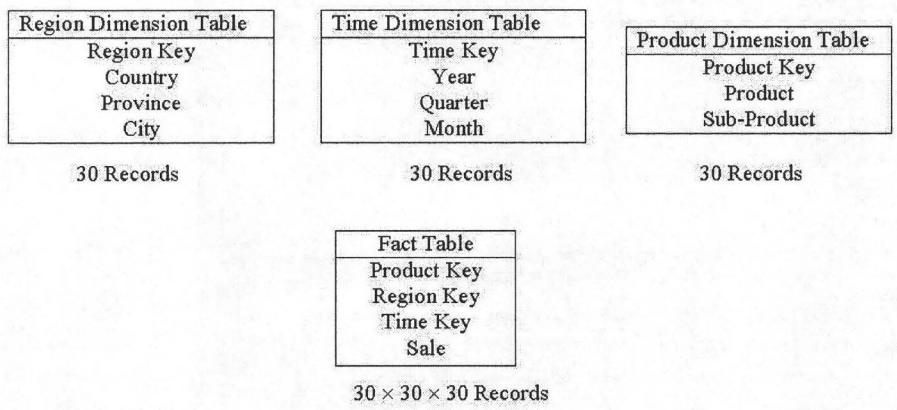


Figure 4.14: ROLAP Space

Compared to the fact table, the space requirement of the dimension tables is negligible.

We will use the same example to calculate the space needs in the multidimensional database for a comparison in the next sections.

4.5.2 MOLAP

In MOLAP, the data is stored in a multidimensional cube structure, an array storage that is inherently multidimensional. Such a design could answer OLAP queries immediately by a specialized storage retrieval – a multidimensional database. To maintain the efficiency and performance of the OLAP queries, an application will usually pre-calculate data and store those results in advance according to requirements. Once a cube is generated, the data is ready for querying. In essence, MOLAP stores big multidimensional arrays and saves them on the disk in advance for future querying. This is a distinct strategy in MOLAP: sacrificing space to speed up queries. Comparing MOLAP and ROLAP, one could see that in the former the required joins

between tables are already available, while in the latter these joins are computed during querying. Moreover, MOLAP approach gives excellent performance parameters for analytic queries since the needed operations along a dimension may be applied quickly.

However, MOLAP also has some disadvantages. It is not always possible or easy to get all data prepared in advance since it the space requirements may too large. Furthermore, some data are seldom required by users, but they would still occupy the resources.

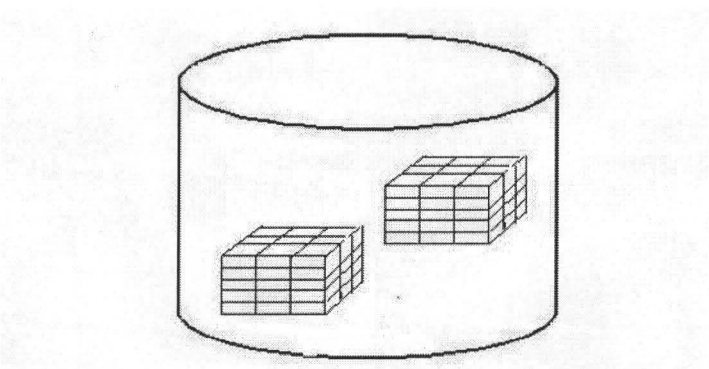


Figure 4.15: Multidimensional Database

Figure 4.15 illustrates the fact that in contrast to RDB in which data are organized into tables, data in a multidimensional database are organized and stored in order to achieve fast navigating performance.

MOLAP Server Architecture

The purpose of MOLAP is to quickly provide analytical results to end-user. It is achieved by preprocessing the data and storing the results in MDDB in advance of querying. When the end-user needs analytical information, the system can provide

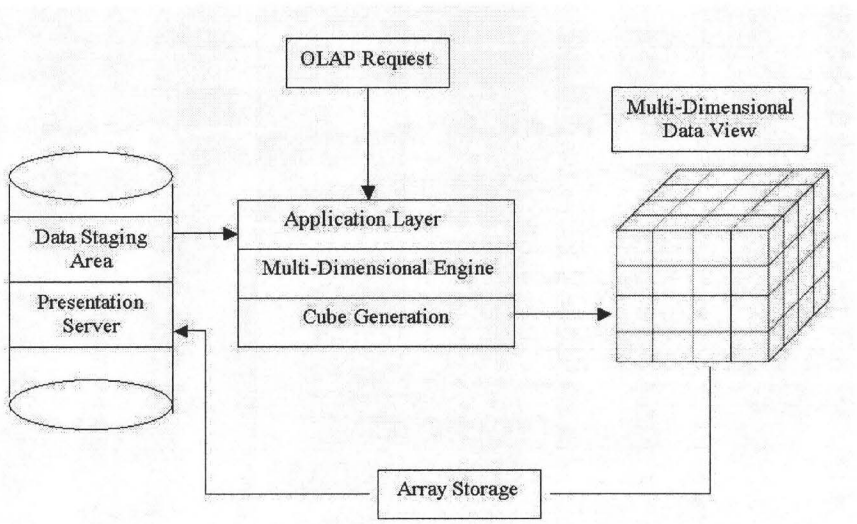


Figure 4.16: MOLAP Architecture

the result directly. MOLAP provides quick response time and good performance, but it does so by sacrificing the space. This phenomenon exemplifies the simple rule: any system performance improvement is typically at the expense of other resources. The MOLAP server architecture is illustrated by Figure 4.16.

We will use the same example used previously to illustrate the space requirements in ROLAP (see Figure 4.14) and demonstrate how to generate data for MDDB and check the space required.

We just use time dimension and region dimension for a simple reason. There are three levels in the time dimension: Year, Quarter, and Month and three levels in the region dimension: Country, Province, and City. Through different combinations, we will create all multidimensional tables which are used to generate the data cubes in MDDB. From aggregated to detail along each dimension (Country – Province – City;

Year – Quarter – Month), one 2-dimensional object with 3 levels in each dimension will generate 9 multidimensional tables as the Figure 4.17 shows. They represent the different dimensions and levels of information from the end-user's view. As we can see, the table 9 is the most detailed containing the lowest level of information.

Table	Country	Province	City	Year	Quarter	Month	Sale
1	√			√			√
2	√			√	√		√
3	√			√	√	√	√
4	√	√		√			√
5	√	√		√	√		√
6	√	√		√	√	√	√
7	√	√	√	√			√
8	√	√	√	√	√		√
9	√	√	√	√	√	√	√

Figure 4.17: Combination Table

There are some other combination tables which do not follow strictly from-aggregated-to-detail rule. We list some examples in the Figure 4.18. They could be derived from table 9 or other tables listed above according to the business requirement, although some of them are meaningless in reality.

From the above multidimensional tables, we generate data cubes for MDDB. Some applications would like to keep all combinations of data cubes for every level. However, depending on the actual business requirements, we may decide whether to generate all cubes. The pre-calculation takes time and the results need be stored in a presentation server. To find a balance between space and performance, we typically store the results of the most frequent requests into a MDDB and generate others from most detailed cube (table 9) or other cubes when the end-user sends a request.

Table	Country	Province	City	Year	Quarter	Month	Sale
1		√		√			√
2		√		√	√		√
3		√		√	√	√	√
...	...	...	...	...	...	...	...
...	√		√	√			√
...	√		√	√	√		√
...	√		√	√	√	√	√
...	...	...	...	...	...	...	...
...	√				√		√
...	√	√			√		√
...	√	√	√		√		√
...	...	...	...	...	...	...	...

Figure 4.18: Other Combination Table

MOLAP Space

Like for the example for ROLAP server, we assume there are the time, the region, and the product dimensions and that there are 30 tuples in each dimension. Each column length is 10 bytes. There is only one measure named "Sales". We can determine the space needs for the most detailed table (table 9), see Frigure 4.19.

Table 9: 30 * 30 * 30 * 9 * 10 = 2430000 bytes

We can see that even just 1 table needs much bigger space than all space required in ROLAP. Note that there are a couple of other aggregated tables to be generated. Furthermore, we will need some temporary space when we generate other tables from the most detailed table.

Since different MDDB vendors may use different data array arrangements, the actual data requirements could vary significantly. However, we just use the sample calculations to illustrate that MOLAP in general requires more space than ROLAP.

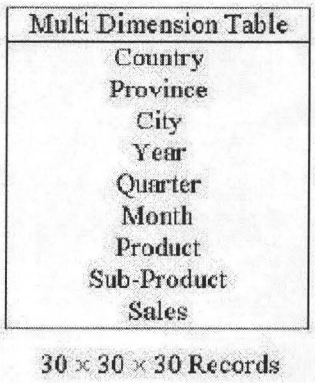


Figure 4.19: MOLAP Space

In conclusion, MOLAP needs a large amount of space and preparation time in exchange for the better analytic performance.

4.5.3 HOLAP

In a HOLAP design, the cube technology is used to achieve a fast performance from summarized information. Once we need the detailed information, HOLAP is able to "drill through" from the cube to underlying relational data. The relational data could be in the data staging area or inside the presentation server.

In an effective HOLAP design system, MOLAP should be able to cover directly at least 70% of the most frequently used business queries. The remaining requests will be handled by ROLAP at times when users send out the actual queries.

In general, the space required for a HOLAP architecture (see Figure 4.20) is less than the MOLAP approach, but the speed of queries in HOLAP is better than that in ROLAP. Typically, HOLAP's performance lies somewhere between MOLAP and

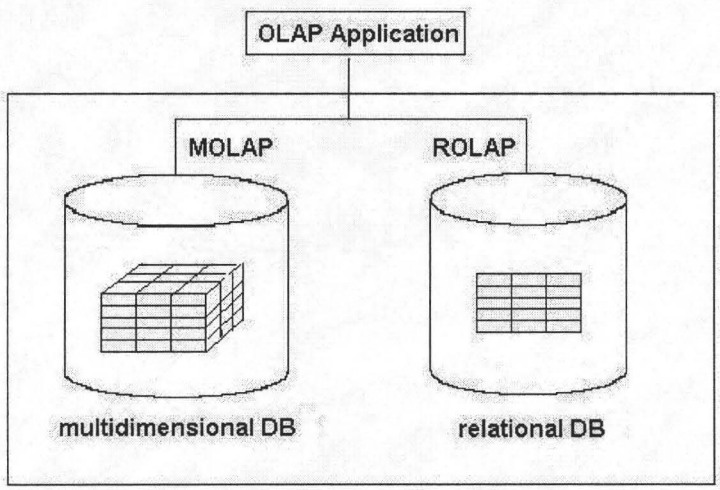


Figure 4.20: HOLAP

ROLAP, but both the technical skills and the complexities involved to implement a HOLAP application are much higher than either ROLAP or MOLAP.

4.5.4 ROLAP vs. MOLAP summary

Ever since the OLAP industry got started, there has been an ongoing debate about the best way to store multidimensional data. The choice of OLAP affects data organization, structure, storage, and system design. Obviously, both mechanisms have their merits as we can see both approaches are widely used by many applications.

In a MOLAP design, data is stored in array storage which makes data inherently multidimensional. It means that data in MOLAP server are stored multidimensionally and viewed multidimensionally. MOLAP approach is easy to design and navigate, it simplifies the maintenance of data, especially for a complex analytical systems. However, MOLAP system is much more complicated than a traditional database

system. The view points are comparatively fixed due to the pre-set cube dimensions. As a relatively new technology, it usually needs more funding for training and system maintenance. One of the challenges of MOLAP is the data growth that affects the number of dimensions and levels in cube hierarchy. With the increased number of dimensions, the number of data cells will expand at an exponential rate.

There is no space limitation for a ROLAP method. Relational database technique is a mature technology and all related skills can be used. Without the physical multidimensional cubes, ROLAP is able to handle more dimensions than MOLAP. Furthermore, there is no fixed number of dimensions for analysis. However, ROLAP performance is worse than MOLAP although most database vendors have optimized their products for OLAP functions which need fast query performance for a large data set. It is hard to design and maintain a ROLAP system.

It really depends on the type of applications to decide which method is more suitable. Most businesses will have to invest in both technologies. Generally, ROLAP is good for simple analysis and reporting needs while MOLAP is better for specialized and advanced analysis in a business planning system.

Chapter 5

Data Process

Data process is an important and complicated aspect of any data warehouse project. It defines and/or describes how the data are extracted, collected, transformed, and stored in the data warehouse.

A data warehouse is a large database that integrates data from multiple and heterogeneous sources. There is a wide range of data sources: operational databases, spreadsheets, text files, and any other data sources. However, mostly transactional data from operational databases are the source of data for data warehouse. For analytical purposes, the transactional data need to go through a series of steps in the data processes: extract, transform, and load (or ETL in short) to provide a bases for a useful analysis. Though software vendors offer a wide range of effective ETL tools, sometimes special requirements apply to the ETL process. In this chapter, we will briefly discuss the main steps of the data process.

5.1 Extract Data from Data Source

After the business requirement is clearly specified, the first task is to figure out what and where we can find those data needed for the efficient analysis in the operational

systems within the organization. Data extract phase will take all useful information from possible sources and keep it in a temporary storage for future data cleansing. The owners of operational systems seldom allow data warehouse programmer the direct access to their systems to retrieve data. To minimize security risks and performance impact in source systems, most operational systems will only provide periodical data extracts and generate flat format files.

Operational system developers need to create programs to extract certain fields and tuples from the source system since they do not want data warehouse programmers to touch their system. If an operational system allows external access, data warehousing system developers can develop some interfaces to extract necessary data by ODBC or other techniques. No matter what approaches we take, data warehouse designers and final users need to involve the extract process very closely with the operational system developers. It will guarantee the quality of the source data for the data warehouse. In most situations, two kinds of scripts should be provided: the **initial extract script** is a one-time-use script which gets all historic data before a data warehousing system is put into a production environment. The second script is an *incremental extract script* which will be run periodically. The incremental extract script is able to collect all new or updated records after the last extraction.

The next critical factor to be determined is when and how often data should be extracted from the operational systems. In most cases, we will select a time period after the business is finished to extract business data, such as the end of day or weekend. Different operational systems could decide different extract time and the extracted data storage.

Operational database schema could change anytime during the data warehousing system development, or even after the data warehousing system is already put into production. Because of these concurrent or subsequent changes, both the operational system developers and the data warehouse designers need to work together and check whether the changes will affect data warehouse analytical function, then make appropriate adjustments if necessary.

5.2 Transform Data into Global Schema

After the data are extracted from the operational systems, the ownership is shifted to the data warehouse management. Usually the extracted data are kept in a temporary storage - so-called *data staging area*. Before we load data into the data warehouse, we need to implement a series of data processes first. The data from a data source needs to be cleaned and integrated. A data warehousing system could have several data sources. There is a lot of incoherence among the sources. For example, an object may appear under different names, or with different data formats in various systems although they represent the same object. In order to solve these conflicts, a global model schema has to be defined. Duplicate records must be removed and some missing values must be provided within the process. In short, we need to transform all raw data from different sources into a global and unique schema representation which contains clean, valid, and quality data.

5.2.1 Integrate

It is a very challenging task to integrate data from multiple sources since different applications may have various definition, standards, and data format. They may use

different methodologies, operating systems, and databases. Different applications are developed by different software companies. All these factors make data integration particularly troublesome. The goal is to combine data from different sources into one global schema database.

5.2.2 Clean Data

The data must be cleansed before they are added into a data warehouse. Obviously, poor data quality is detrimental to the accuracy of the future business analysis.

Below we will discuss some common problems encountered during the data cleansing stage. It is common that many unexpected and unanticipated problems show up during cleansing.

1. Different data standards:

   For instance, an address like "1025 Victoria Park Avenue, ON, M3W 2W6, Canada" could be represented in many different ways. The word "Canada" may also be stored as "CA", "CAN" or "CANADA". The postal code "M3W 2W6" may be represented as "m3w 2w6" or" M3W2W6". We need to decide a standard format to build data for a data warehouse, and search and change these values to the desired format.

2. Different data lengths:

   Two fields of same meaning in different systems may be defined in different lengths. We also need to define a data standard in order to cut the longer one or extend the shorter one.

3. Duplicate record elimination:

   For example, a data warehouse for retail banking extracts data from both the

saving and the checking database. Both databases have a customer named "Paul Louis" with the same SIN number. All other information are exactly same. We should not generate two tuples.

4. Duplicate record merging:

   It is quite common, that two data sources contain the same object, but each of them has incomplete information concerning the object. In such a case, we need to merge the two records and make a new record with complete information .

5. Missing data:

   Some records may have some data (fields) missing. For major fields, the data warehouse developer must validate the data and supply the missing data.

There are possibly many other types of "dirty" data. In general, this process is very messy and very time consuming, and often quite ad-hoc. The cleansing strategy has to be frequently revised according to the new issues occurring.

5.2.3 Transform Data

As we mentioned in Chapter 4 regarding OLAP, we need to perform some pre-calculations for later analysis. The new values need to be derived from the extracted data. The levels of granularity need to be considered first. Most applications uses *lookup rules*. The data will be assembled for dimensional and fact tables according the these rules. Data aggregation may be a part of the transformation process. Most data warehouses only keep aggregated data in the presentation server.

5.3 Load Data

The last step of the data process is to load the clean, integrated, and summarized data from the data staging area into the presentation server. Data loading is also a time-consuming job. Heightened caution is required before providing the final data to users. We list a few basic guidelines:

1. It is useful though expensive to create some software facilities to check data integrity and quality. It warrants the accuracy of information for future analysis.
2. Usually data warehouse is a large database. To improve the analytical process performance, data should be indexed and partitioned.
3. If data marts are based on a warehouse, they need to be refreshed at the same time when data is loaded into the warehouse.

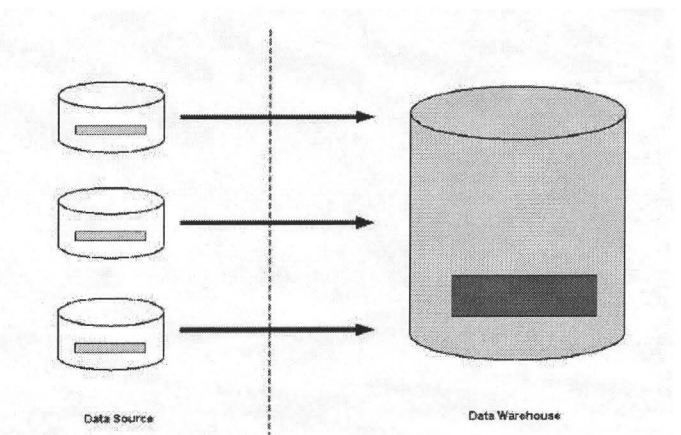


Figure 5.1: Initial Data Loading

5.3.1 Initial Data Loading

All historic data must be loaded when the data warehousing system is implemented and put into the production environment. Usually, the size of the initial load is quite large although it is a one-time job (see Figure 5.1).

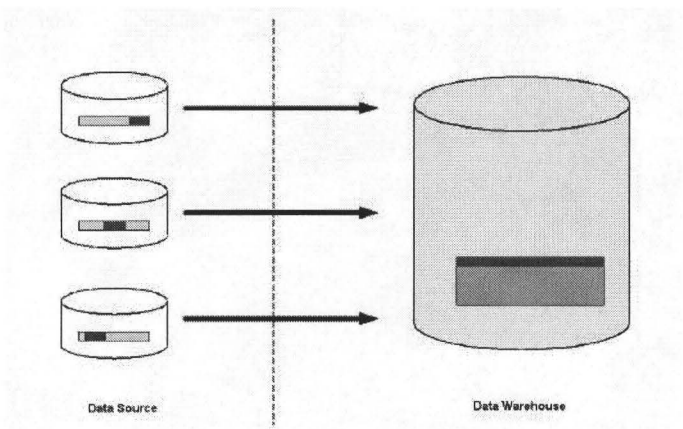


Figure 5.2: Incremental Data Loading

5.3.2 Incremental Data Loading

It is necessary to determine how often the incremental data should be added. The new and changed data will be extracted from data sources and appended into the data warehouse. Unlike the initial data load, the size of the incremental data is much smaller (see Figure 5.2).

5.4 Data Lifetime

Although the data in a data warehouse are usually read-only, they have to have a limited period of usefulness. In any case, if the data were stored indefinitely, the data

warehouse size would expand beyond manageable. At some point, old data needs to purged from the data warehouse. However, it is not as simple as removing the outdated records from the data warehouse. In most cases, the outdated data will be moved to an external storage, such as tape, and archived. Then the detailed data will be rolled up to a higher summary level before it is purged. For example, we summarize the daily data to the weekly or monthly level first, then we purge the daily data from the data warehouse.

Chapter 6

A Case Study — S Bank Data Warehouse

A disclaimer: This case study is based on a data warehouse project of S Bank of Shanghai, China. We have S Bank's permission to use the project in this thesis as a case study in design and implementation of data warehouse. However, for obvious reasons, the data used in examples are not the original proprietary data of S Bank, but data made up just for this thesis.

6.1 S Bank Background

S Bank^{TM} is one of the largest banks with a good customer resource in China. S Bank's headquarters are located in Shanghai and it has branches in most provinces of China. There are approximately 17 million people in Shanghai. More than half of that population are S Bank's customers. Considering the large population in the whole country, we can imagine how much new data will be generated every day. In China, banking system must separate individual and company businesses according to the government policy.

S Bank's IT system consists of several separate banking systems, credit card systems, trading systems, management systems, risk control systems, and other systems. They use various operating systems and databases. Operating systems used include Unix™, Windows™, Tandem™ mainframe, while the existing databases are based on Informix™, Sybase™, Oracle™, and MS SQL Server™ respectively. In one particular case the odd UNIFY™system (a file format system) is used. However, most systems are based on the client-server architecture and store data in the headquarters data center. The data grow extremely fast due to the large number of customers. For instance, one individual transaction system could generate approximately 3 Gb of transaction data per week.

There are several similar transaction systems for different businesses in S Bank. To build an enterprise level data warehousing system based on such big data resource is a great challenge, but may also bring substantial benefits with the efforts.

6.2 Foreign Currency Trading System

Foreign Currency Trading System of S Bank (FCTS for short) is a real-time system which provides services for customers to sell one currency and buy another. FCTS is similar to a stock exchange system, but trades currency instead of stock. Reuters News provides fresh currency trading quotes. S Bank receives fresh quotes from rented satellite channel provided by Reuters News. There could be several patches of quotes from Router News in one minute. According to the quotes, S Bank will generate its own bid and ask prices in time. In order to be profitable, S Bank uses a higher price to sell and a lower price to buy a currency. For the customers, they will trade different currencies for investment depending on their knowledge. Unlike stock

exchange system which is only open for 8 hours a day, FCTS needs to run about 20 hours every day to work with the Asian, European, and American currency markets. It is a system that requires a long running time and very fast response.

FCTS servers use SCO UNIX OPEN SERVER 5.04 platform and SYBASE Adaptive Server 11.0.3 which are located in two Compaq servers using a warm backup software. Each server has a dual CPU, RAID 5-disk array, and 4 Gb main memory. There are several client applications based on both Unix and Windows. The software used includes APT, SQR, Delphi, Visual Basic, and C++.

The overall architecture of FCTS is shown in Figure 6.1.

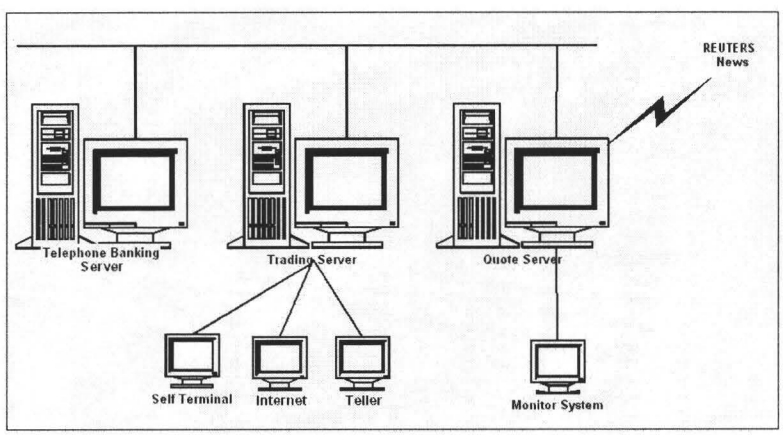


Figure 6.1: Foreign Currency Trading System

FCTS transaction flow is given in Figure 6.2.

FCTS provides four major functions:

1. Customer management function

   Customer management module keeps basic customer information. There is an

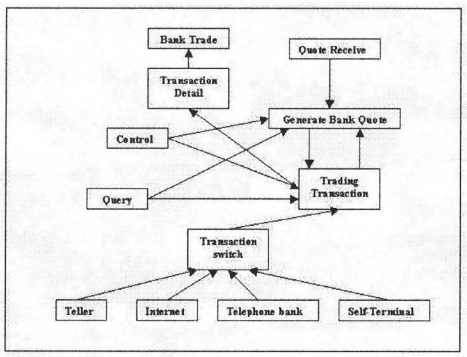


Figure 6.2: Foreign Currency Trading System Transaction Flow

interface between FCTS and Customer Management System (CMS) if more detailed information is needed. CMS in S Bank includes all customer information, such as customer's age, profession, income, and credit record, etc. It will provide sufficient information for the customer dimension in a data warehousing system.

2. Banking system function

   FCTS contains all banking system functions: open/close account, debit/credit, balance inquiry, etc. We can obtain some basic customer financial information from this component. S Bank classifies several types of customer groups based on several factors, such as income, credit, deposit, and transaction amount/volume in S Bank.

3. Trading function

   Trading function is the core of FCTS. There are four transaction sources: teller, self-terminal, telephone, and internet. Customers can place instant transactions according to the current quote. They could also place transactions through

an agent - at a price they wish to buy or sell a certain currency. Different price quotes will be given according to a customer's class during the trading transaction.

There are 11 different currencies currently traded in the system. However, S Bank only allows 55 combinations of from/to currencies such as USD/CAD, EUR/JPY... It will be the sources of money combinations dimension in a data warehouse. (Note that the project described here preceded the conversion of many European countries to the common currency of Euro in January 2002.)

4. Control function

   This module provides monitoring and manual interception. S Bank can terminate a quote and stop the transaction if there is an unexpected problem. The module does not provide any useful resource for data for a data warehousing system. The related data will not be included in an ETL model.

After accumulating some amount of trading currency, S Bank will try to find a good time to sell it. At the same time, if S Bank feels it is risky to hold a certain currency since market changes sharply, they will sell it immediately. Unlike an individual transaction with a small amount of money, there is a minimum amount required equal to 100,000 USD for each trade placed by a bank. The higher the amount of the transaction, the better the trade quote S Bank could get. For such a large transaction, the analysis in advance is very important to catch the right time to place a transaction order. Thus, an analytical function is needed to provide quote forecast according to historical information for a manager to make decision. The competition is very high since there are more than 20 banks involved in such a business. To attract customers, most banks provide all kinds of convenient trading tools and services. In order to keep

old customers and attract new investors, S Bank collects information about possible customers and increases the facilities and services it can provide to its customers. At the same time, they involved in some risky investments since a customer can borrow money (on margin) from the bank for the trading system. S Bank's top management thus decided to establish an enterprise data warehousing system to cover all of these analytical and other related requirements in a short period.

6.3 Data Warehouse for FCTS

S Bank began to design the first data warehousing system in 1998. It experienced a major change in 2000. The bank got different requirements from individual departments. However, there was a weak relationships among different department's data and requirements. Thus, S Bank data warehouse system was based on enterprise data marts architecture. Different data marts are implemented in production for different businesses. They could be implemented at different time without impacting others. The whole enterprise data warehousing system was finished in 2002. Based on this architecture, more data marts could be added and upgraded flexibly in the future.

S Bank's data warehousing provided analytical functions for several business systems. However, as the whole system is too large to be described in this thesis, we focus on a data mart for FCTS. In this case study, we will demonstrate several important component implementations: ETL, system design, and data model.

6.3.1 Data Acquisition

After all business requirements are specified clearly, the first task is to decide what data needs to be extracted from existing data sources. We should only include useful columns in the extraction process. Unnecessary column may not affect the data warehouse storage too much initially. However, it is important to note that data warehouse will keep all historic data and expand extremely fast. Storage space problem will become one of the critical problems after data warehouse is implemented and in production.

Once the data sources are determined, the selected columns are marked and extract strategy is decided. Figure 6.3 shows an extract strategy example for FCTS data warehousing system.

DW_Extract

	A	B	C	D	E	F	G
1	Source System	Source Table name	Column	Type	Extract	Extract Strategy	Subject
2	FCTS	hist_quote	branchid	varchar(4)	N		
3	FCTS	hist_quote	combno	smallint	N		
4	FCTS	hist_quote	mountbench	float	Y	FULL	T99
5	FCTS	hist_quote	pointbase	int	N		
6	FCTS	hist_quote	remitbuy	float	Y	FULL	T99
7	FCTS	hist_quote	cashbuy	float	Y	FULL	T99
8	FCTS	hist_quote	remitsale	float	Y	FULL	T99
9	FCTS	hist_quote	cashsale	float	Y	FULL	T99
10	FCTS	hist_quote	quotedate	int	Y	FULL	T99
11	FCTS	hist_quote	quotetime	int	Y	FULL	T99
12	FCTS	hist_quote	quotestatus	tinyint	N		
13	FCTS	hist_quote	quoteflag	char(1)	N		
14	FCTS	hist_quote	quoteser	int	N		
15	FCTS	hist_quote	sendflag	tinyint	N		
16	FCTS	hist_quote	stopflag	tinyint	N		
17	FCTS	hist_quote	quotesource	tinyint	N		
18	FCTS	hist_quote	serialno	smallint	N		

Cover / SUM / Sheet3

Figure 6.3: Data Extract

Like most applications, FCTS developers do not want the data warehouse group developers to access their database since FCTS is a banking and trading system which

requires extremely high security. FCTS developers will be responsible to provide the extracted data and create the data files under a pre-agreed format.

Nowadays, there are many ETL tools available. However, in 1998 ETL tools were limited and their functions were not as versatile. Even now, most versatile ETL tools cannot satisfy all requirements of every system. Accordingly, we have to do some programming for data extraction. For FCTS data warehousing project, most extracted data are generated by custom-made programs.

The table `his_quote` provides all quotes in FCTS. Data warehouse needs to keep all historic quotes to analyze and forecast the currency trends. Below is an example of a function to extract quote later than a given date. It provides incrementally fresh quote for FCTS data warehousing system.

```
*********************************************************************
int DbDumpHistQuote(int pi_Date) {
    FILE *fp;
    char buf[50];
    DBCHAR s_Note[21],s_TrTime[7];
    DBSMALLINT i_MoneyNo1,i_MoneyNo2;
    DBINT i_TrDate;
    int i_BcpLine,i=0;
    double atof();
    DBFLT8 f_CashBuy,f_CashSale,f_RemitBuy,f_RemitSale,f_MountBench;

    if (dbcancel(_p_DbProc) == FAIL)
        return -1;
    dbcmd(_p_DbProc, " SELECT a.moneyno1, a.moneyno2, mountbench,remitbuy,");
    dbcmd(_p_DbProc, " cashbuy,remitsale,cashsale,quotedate,quotetime");
    dbcmd(_p_DbProc, " FROM   febmain..money_comb a,febhist..hist_quoteb ");
    dbcmd(_p_DbProc, " WHERE  a.combno = b.combno");
    dbfcmd(_p_DbProc, " and    b.quotedate \>= %d ",pi_Date);
    dbsqlexec(_p_DbProc);

    if( dbresults(_p_DbProc) != SUCCEED ) return -1;

    dbbind(_p_DbProc, 1, SMALLBIND, (DBINT)0, (BYTE *)&i_MoneyNo1);
    dbbind(_p_DbProc, 2, SMALLBIND, (DBINT)0, (BYTE *)&i_MoneyNo2);
    dbbind(_p_DbProc, 3, FLT8BIND, (DBINT)0, (BYTE *)&f_MountBench);
    dbbind(_p_DbProc, 4, FLT8BIND, (DBINT)0, (BYTE *)&f_RemitBuy);
    dbbind(_p_DbProc, 5, FLT8BIND, (DBINT)0, (BYTE *)&f_CashBuy);
```

```
    dbbind(_p_DbProc, 6, FLT8BIND, (DBINT)0, (BYTE *)&f_RemitSale);
    dbbind(_p_DbProc, 7, FLT8BIND, (DBINT)0, (BYTE *)&f_CashSale);
    dbbind(_p_DbProc, 8, INTBIND, (DBINT)0, (BYTE *)&i_TrDate);
    dbbind(_p_DbProc, 9, NTBSTRINGBIND, (DBINT)0, (BYTE *)s_TrTime);


    fp = fopen("/usr/DW_FTP/T99_price.txt","w");
    while ( dbnextrow(_p_DbProc) != NO_MORE_ROWS )
    {
        fprintf(fp,"%d %d %8.4f %8.4f %8.4f %8.4f %8.4f %d %s\n",
            i_MoneyNo1,i_MoneyNo2,f_MountBench,f_RemitBuy,
            f_CashBuy,f_RemitSale,f_CashSale,i_TrDate,s_TrTime);
    }
    fclose(fp);

    return CHECK_SUCCESS;
}
*********************************************************************
```

6.3.2 Data Transformation and Aggregation

As discussed previously, operational databases keep individual records, but a data warehouse only keeps aggregated data. After transaction data are extracted, they need to go through a transformation. The transformation process includes file validation, data cleanse, data integration, consistency check, de-normalization and re-normalization, data aggregation, and data quality assurance. Not every application needs all these steps. Data source quality will determine which steps need to be performed. The example in Figure 6.4 is a transformation and aggregation strategy in FCTS data warehouse project.

Actually, no matter what OLAP approach we take, the data will be pre-calculated and stored into a relational database or a multidimensional database. Let us recall that MOLAP may save every level data into a cube after the lowest level (grain) is decided while ROLAP only saves a couple of levels of results and calculates other levels of results from them. Also, MOLAP can provide the analytical data directly

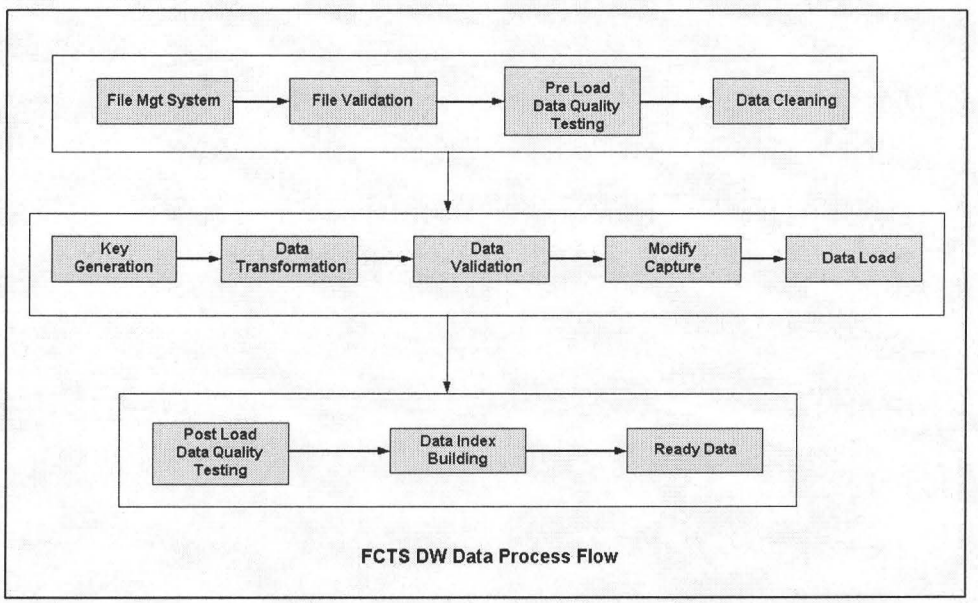


Figure 6.4: ETL Process Flow

from the cube while ROLAP needs to get results from joined tables.

In Figure 6.5 are the extracted records from one table in the FCTS operational database.

Most applications use a day as the basic unit for time. A day unit could usually satisfy normal analytical functions. However, due to the special characteristics of a trading system, such as a stock exchanging system or currency trading system, more detailed analytical unit is needed since the quote could change a lot in just 1 or 2 minutes. So using a day as a unit is neither accurate nor sufficient for the analytical system. Due to the special feature, we will separate the date and time dimensions along time feature. As a result, there are three dimensions for this table: money combinations dimension, date dimension and time dimension. The lowest level

MoneyNo1	MoneyNo2	Mount Bench	Remit Buy	Cash Buy	Remit Sale	Cash Sale	TrDate	TrTime
USD	JPY	115.685	115.665	115.665	115.705	115.705	20060616	00:29:32
...	...	...	...	...	...	...	...	...
USD	JPY	115.678	115.658	115.658	115.695	115.695	20060616	00:22:21
USD	JPY	115.648	115.628	115.628	115.675	115.675	20060616	00:21:29
...	...	...	...	...	...	...	...	...
USD	CAD	1.1125	1.1123	1.1123	1.1127	1.1127	20060616	00:20:29

Figure 6.5: Extracted Record

(grain) of time dimension is 5-minute interval.

Each application needs to determine some transformation rules for data conversion. Transformation process will go through all records and generate surrogate keys according to the business rules. For money combinations dimension, since the combination will not change, it is a static dimension. We assign a money combinations key to them, such as USD/JPY (1), USD/CAD (2)... For time column, since we use 5 minutes as the lowest level, we make a rule such as assigning 00:00:00 - 00:04:59 as (1), 00:05:00 - 00:09:59 as (2), ... , 18:00:00 - 18:04:59 as (216) ... Date dimension will work in the same fashion. The above table will be converted to the one given in Figure 6.6.

Combo Key	Date Key	Time Key	Mount Bench	Remit Buy	Cash Buy	Remit Sale	Cash Sale
1	167	6	115.685	115.665	115.665	115.705	115.705
...			...	...	...	...	...
1		5	115.678	115.658	115.658	115.695	115.695
1		5	115.648	115.628	115.628	115.675	115.675
...		...	...	...	...	...	...
2	167	5	1.1125	1.1123	1.1123	1.1127	1.1127

Figure 6.6: Generate Unique Key For Each Entry

After the conversion is finished, the transformation process will look up the records. It separates the records and generates data files for fact and dimension tables. At the same time, some useful information will be derived during this process. For example, in the data dimension, Jan 3rd belongs to Week 1, Quarter 1, etc. For different combinations of dimensions, transformation process will calculate the measurement valve according to the requirements for fact table group by the different scopes. For instance, start, end and average quote of Mount Bench, Remit Buy, Cash Buy, Remit Sale, Cash Sale.

Finally, we will get aggregation data files ready for loading. There is no individual information there. A couple of records will be treated as a group belonging to a certain scope.

Combo Key	Date Key	Time Key	Mount Bench	Start Remit Buy	End Remit Buy	Average Remit Buy	Start Cash Sale	...	Total Quote
1	1	5							16
...									
1		17							23
1									
...		...							
2	365	8							18

Figure 6.7: Fact Table

In this example (see Figure 6.7), a file is for the fact table. It contains three dimension keys and the measurements such as the start, end and average quotes of Mount Bench, Remit Buy, Cash Buy, Remit Sale, Cash Sale) and also a total quote number in this range.

There are three files for every dimension table (see Figure 6.8). As we can see, there are some derived information in those records, such as the day, week and quarter.

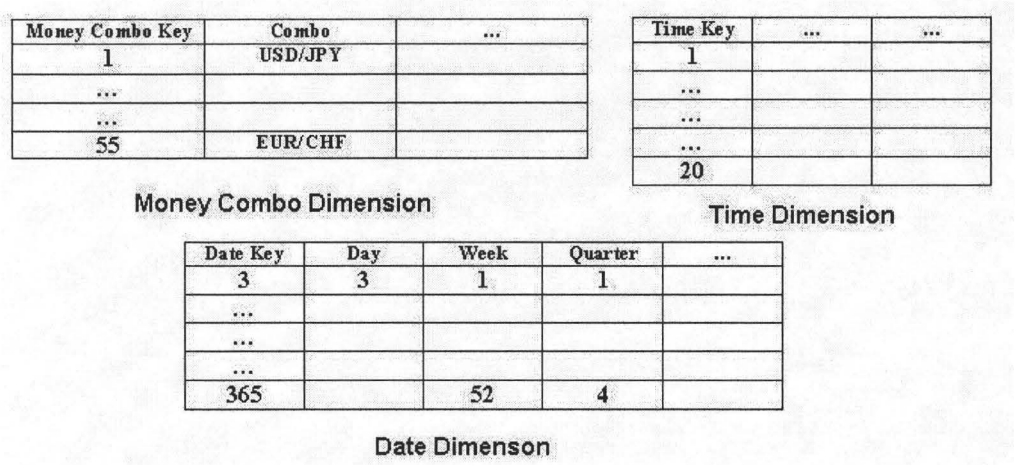


Figure 6.8: Dimension Table

6.3.3 System Design

In 1998 data warehouse as a concept and application was still quite new in China. For most people in S Bank's IT department, this was their first data warehouse project. It is hard to define a successful data warehousing system – the criteria are quite subjective and are related to the resource availability, funding level, and technology used. However, satisfying the needs of end-users is always the first goal.

Phase 1

According to the degree of urgency of business requests, the whole system was separated into several phases. In Phase 1, the data warehouse project group tried to meet with those departments that have urgent requests. At the same time, it was also the first step for the whole enterprise data warehouse system.

The bank's business department gave the detailed requirements about analytical functions and requested that FCTS data warehouse were to be implemented as soon

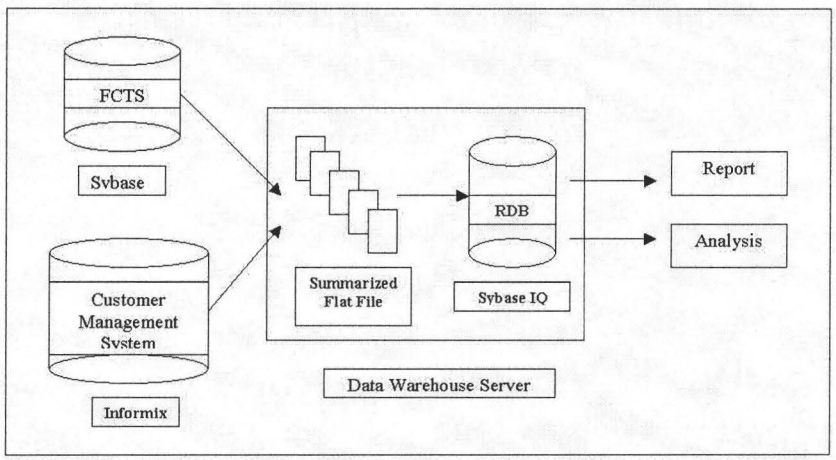


Figure 6.9: FCTS System Design Phase 1

as possible due to critical market competition. Once other banks get the customers, it would be hard to win them back. After extensive and careful research, the IT department proposed a design and believed the existing server could meet the requirements using the design.

The proposed system design is shown in Figure 6.9. There are two data sources, FCTS and CMS. To save space for the data warehouse server, two parts of the ETL process are performed in data source systems. First, useful data are extracted into some temporary tables and text files. The data go through cleansing and integration. After that, data transformation is implemented according to original business requests. The summarized records are generated into the flat format files and transferred to the data warehouse server.

Because the server for the data warehouse project was given before the project started and thus before the space required had been estimated, we decided to adopt a ROLAP approach (to minimize redundancy). After careful research we concluded

that the server space was sufficient for this design. Our design followed the enterprise data marts architecture which was reasonable and practical to finish in the short time that was given to us. The design covered all business department requirements.

After the aggregated files were transferred to the data warehouse server, they were loaded into an RDB which lies in the heart of the design. Aggregated record files is the source of ROLAP which supports analytical end-applications. Most developers involved already have had many years experience with relational databases and thud did not need any additional training. Moreover, S Bank did not have to purchase any additional software for MOLAP server. Lastly, RDB is a mature technology and is capable of handling a large amount of data.

S Bank used an architecture similar to other corporations used at that time. Also, the server did not have space to allocate more data except data warehouse itself. FCTS data warehouse project was finished according to this design and implemented and in production with some other finished data warehouse modules in Phase 1.

Basically, the initial design ran well and did help decision makers in business analysis and trend forecasting. However, some shortcomings showed up after the data expanded and the business requirements changed a couple of months later. Let us briefly discuss them:

1. With increased data, some analytical functions performed satisfactorily, but response times of the others deteriorated far beyond the users' expectations. In the original design, we tried to store as many pre-calculated functions as possible directly in the database, but we could only store the lowest level of data and the most-frequently used functions due to space limitations. We had to find a balance between the space limitation and business requirements. ROLAP

server dynamically executes SQL queries for other functions that are not pre-calculated. For these functions, the response time deteriorates sharply when the size of the fact table reaches a certain limit.

2. Once business departments have new requirements, data warehouse group has to work with operational system developers for the design of new aggregate data format and wait for the aggregated data files. It is difficult to control the development time.

3. In general, data warehouse provides information about a group of objects that has been designed in an agreement between business and IT department. However, sometimes analysts may be interested in more detailed data in the real world and still need to find the individual information for some unique scenario. That is a new request for the data warehouse people. We could not find such information in our server and it was hard to enter the operational system to retrieve it either. Furthermore, it was impossible to drill down to a lower level from the aggregated level for some information.

In conclusion, most user complaints were caused by the requirement changes and the data warehouse hardware limitations. Due to the space limitation, it was impossible for us to keep a copy of the original transaction. This limited the flexibility of the system to add new functions. We concluded that staging data area or other similar idea was necessary. It seems that most of the early early applications of data warehousing had similar experience.

Phase 2

Since S Bank's data warehouse system had provided useful analytical information that had helped develop new business, S Bank decided to put more funding into the data warehousing system to derive even more benefits. Business and IT departments decided to enhance data warehousing system and solve the problems experienced in Phase 1.

There are three major modifications in Phase 2. Firstly, we understood that the space limitation experienced in Phase 1 lead to performance problems. With data dramatically expanding in the data warehousing system, we decided to purchase a new server after calculating the necessary space. We also hoped that the latest server model could improve the data warehousing system performance dramatically. Secondly, we decided to keep the original operational data in the new data warehouse server. In a separate schema to the production data warehouse, staging tables would be created. Extracted data from the source systems would initially be loaded to the staging area, where preliminary checks and manipulations could be done without affecting the rest of the data warehouse. Instead of getting the aggregate data files from the operational system, data transformation would be finished in the data warehouse server. This would bring convenience and flexibility to the users who had new analytical requirements or requested specialized information. Lastly, we considered using the MOLAP approach for Phase 2 to improve the analytical querying performance.

MDDB is specially designed for a multidimensional analysis. By calculating or consolidating transactional data and storing every summarized measure at each hierarchy level in advance, MOLAP server delivers impressive query performance. It has its own strengths of data aggregation, data view, and data storage for data analytical

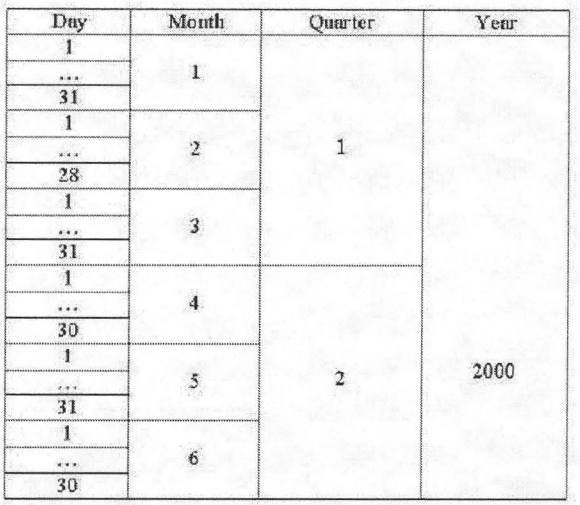


Figure 6.10: Multidimensional Database Hierarchy Summarized

applications. Most MOLAP server products provide utilities to aggregate data along hierarchies. Those utilities also provide fast navigation techniques. As demonstrated in Figure 6.10, MOLAP products can aggregate month, quarter, and year information from the day data automatically.

Although MDDB has many better features for dimensional analysis, we found weaknesses of the MOLAP architecture after a series of tests and intensive research.

1. Theoretically, there is no dimension limit for MOLAP model. However, in reality the analytical performance becomes very slow if the dimension number of a hypercube is more than six. If a cube has more than eight dimensions, the whole system practically halts.

2. According to a published paper [13], MOLAP can only handle rather limited amount of data. Many other studies also shows [14] that the input data should

be less than 200 Gb for a reasonable performance. For comparison, a 50 Mb input would expand to approximately 40 Gb in the final MDDB. Thus, MDDB is better suited for a data mart smaller than 200 Gb input data. As we mentioned previously, S Bank's amount of data was large due to the population served. Obviously MOLAP was not suitable for an enterprise data warehouse server. However, we still considered using it for some modules for the most frequently-used functions.

3. After the cubes are designed, the aggregated data is pre-loaded into MDDB which improves retrieval performance. However, if we change the requirements, sometimes the whole cube may need to be rebuilt completely. It is very time consuming to rebuild the whole data cubes because all hierarchy levels of summarized data must kept in the MDDB.

After we compared strengths and weaknesses of both OLAP mechanisms, we considered two possible designs based on the characteristics of our application.

(1) ROLAP

Because the new server space appeared to be sufficient even for the foreseeable future, we could generate the aggregated data for most levels and load them into the relational database. The prepared data would cover approximately 70% of the frequently required analysis. Business department would be responsible for providing it. This approach would enhance system performance dramatically. However, unlike MOLAP with auto aggregation functions, some aggregation would be generated by additional programming. To avoid utilizing too much storage space, the remaining 30% of functions would be performed during the actual requests.

At the same time, we try to partition database according to date and branch to

achieve better performance. Instead of navigating a large table, it is much quicker to retrieve data from some small tables. Indices would be dropped before data is loaded and then would be rebuilt afterwards. It will speed up data loading and access performance. The expensive and powerful server could compensate OLAP's performance.

The system architecture we arrived to is illustrated in Figure 6.11.

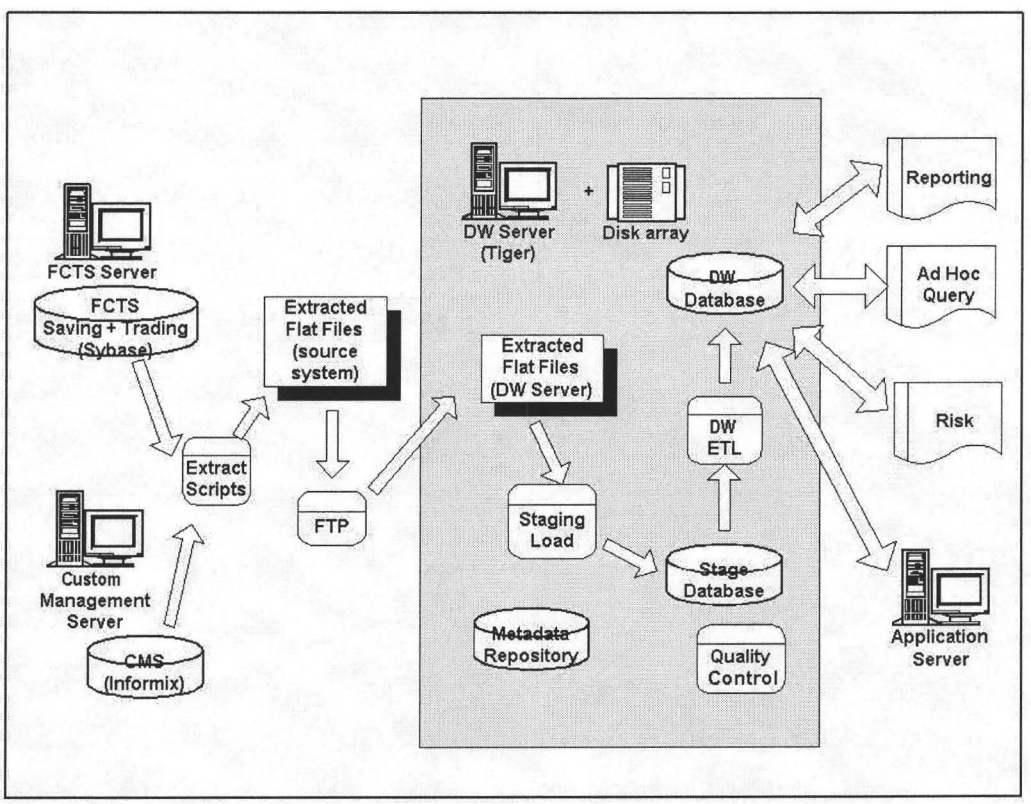


Figure 6.11: FCTS Data Warehouse System Design Phase 2

(2) HOLAP

We would set up both RDB and MDDB on the same server. In other words, the

aggregated data from data staging area would feed both RDB and MDDB.

Since MDDB cannot handle a large amount of data, we would only store the most-frequently used modules in MDDB with 6 months of data. The size of data was limited to under 200 Gb. All other data would be stored into a relational database. According to typical business analysis, the chance to access data more than 6 months old was comparatively low. So we decided that data more than 6 months old would be moved to RDB when new data arrived.

We needed to purchase new MDDB software as well as new MOLAP client tools. Oracle™, Arbor Essbase™ and SAS™ were all considered. Extra training was required for developers. However, HOLAP was a new concept in the academic research only at that time, so we failed to find a real working example in the industry.

Figure 6.12 indicates the difference of the new design in comparison with the first design for Phase 2 (see Figure 6.11).

Finally we decided to follow the first design for Phase 2 after a careful comparison. Not only does MOLAP design have an issue with the size of data, the need to be move data from MDDB to RDB later may cause data inconsistencies. New interface would have to be developed for the data transfer. We would need more funding for HOLAP software and personnel training. At the same time, we concluded that ROLAP approach could be better and more economical for S Bank's data warehousing system. ROLAP mechanism is more suitable for a large data system. We felt confident that we could improve ROLAP performance based on the experience from Phase 1. HOLAP was too new for us and we could not afford another major modification at some later date.

Let us report, that S Bank successfully upgraded the FCTS data mart using the

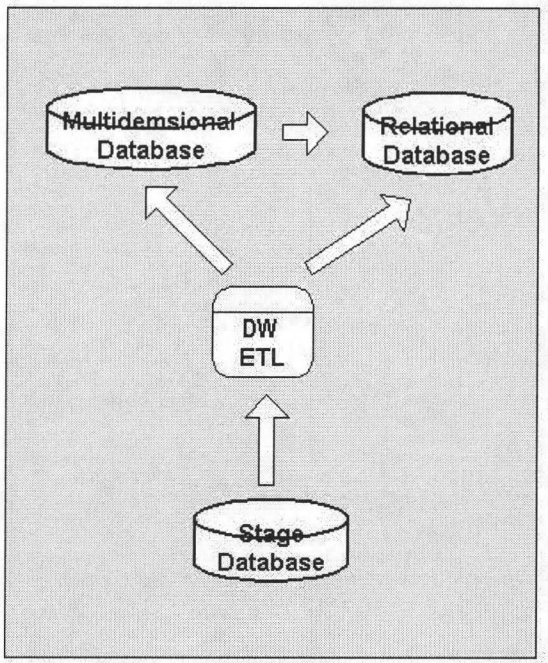


Figure 6.12: FCTS System Design Phase 2 — HOLAP

ROLAP approach in 2001. The success to this day proved that the design we proposed was reasonable, flexible, and expandable.

6.3.4 Data Model

According to textbooks, most data warehouse applications will follow either a dimensional model or a relational model. However, there is no procedure or rule what to choose. It is not impossible to use both models for an application. In any case, the actual implementations vary substantially even if the same data model is used. There are a lot of factors related to data model. The best data model is the one that matches technique, funding, hardware, and strategy in a certain project.

As we mentioned, there are different levels or categories in dimension definition. For instance, we can describe customer dimension by customer's gender, age, credit level, etc. Date dimension could have levels such as day, week, month, quarter, and year. The purpose of category and level is to compare different groups. The problem is where to represent the category and level. We could express them in either a fact or a dimension table. However, a real good design should consider the application data volume, detail business and system operations, etc..

When designing data model for a data warehouse system, we should remember the major differences between data warehouse and operational database. Typically data warehouse is much bigger than operational database. The main operation of data warehouse tools is to retrieve data. Thus the query performance is always the first priority. We allow data redundancy in data warehouse as long as it could help improve access speed.

These were the guidelines we used to investigate various data models for S Bank data warehouse project.

Model 1

S Bank needs to analyze transactions along the time and branch view. We can thus put time and region as attributes into fact table, see Figure 6.13.

It is fast and convenient to get information along a region or a time feature. For instance, it is easy to calculate a city's transaction volume. It does not need another table's information. For small systems and if data is not large this is a rather satisfying model. However, this design will increase the number of attributes in the fact table. If the application data volume becomes big, the fact table will become difficult to manage. Moreover, the fact table must be redesigned and re-constructed if there is

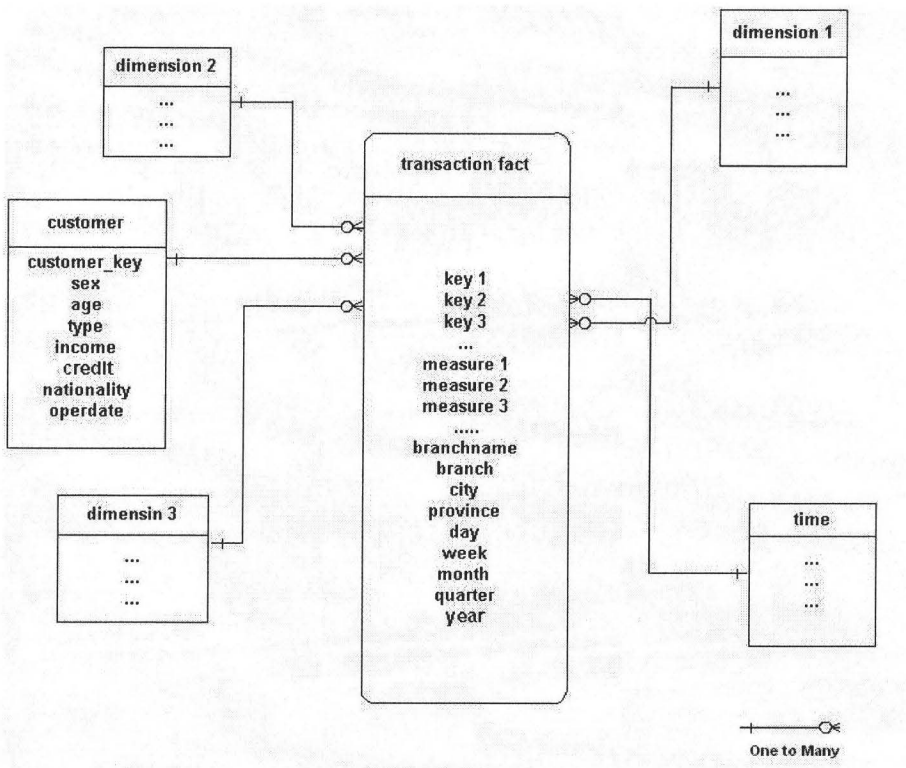


Figure 6.13: Model 1 - Reflect Dimension Information in Fact Table

any change related to the two dimensions. So this design is good for a system which has a small data and static dimensions. Most analytic functions are along the fixed dimensions.

Model 2

We can set date and region dimension table for those attributes. The fact table and dimension tables are connected by surrogate keys, see Figure 6.14.

This design could reduce the number of records in the fact table. Furthermore, once we need to change the dimension information, it only affects the dimension

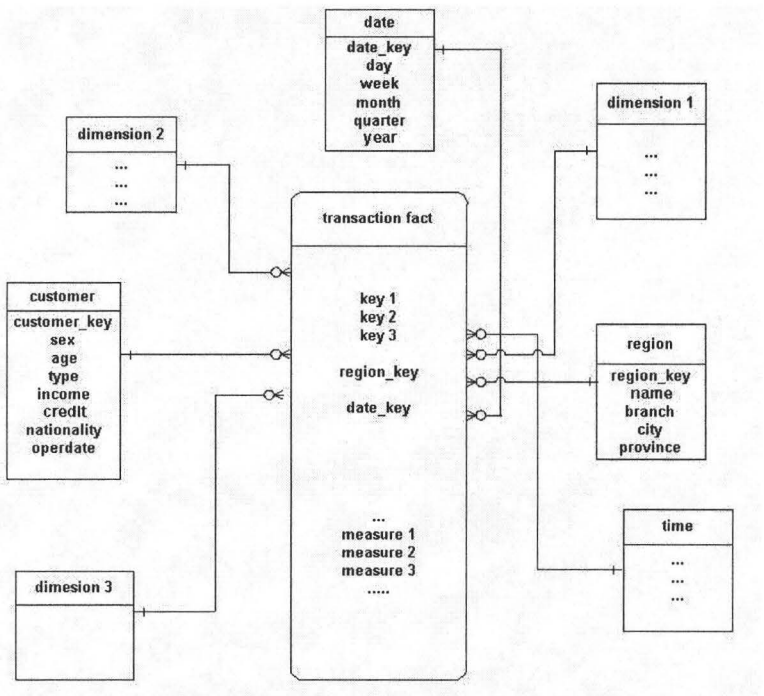


Figure 6.14: Model 2 - Setup Dimension Table

table. It is better for an application with a large volume of data. However, OLAP's performance is worse than in model 1 since we need to join more tables for computing the result.

Model 3

We can reflect the dimension level in a dimension table. We use a join between the fact table and the customer dimension tables. It is a snow flake schema, see Figure 6.15.

This design not only reduces columns and records in the fact table, but also saves space for dimension tables since it reduces redundancy. However, we have to pay more on query performance.

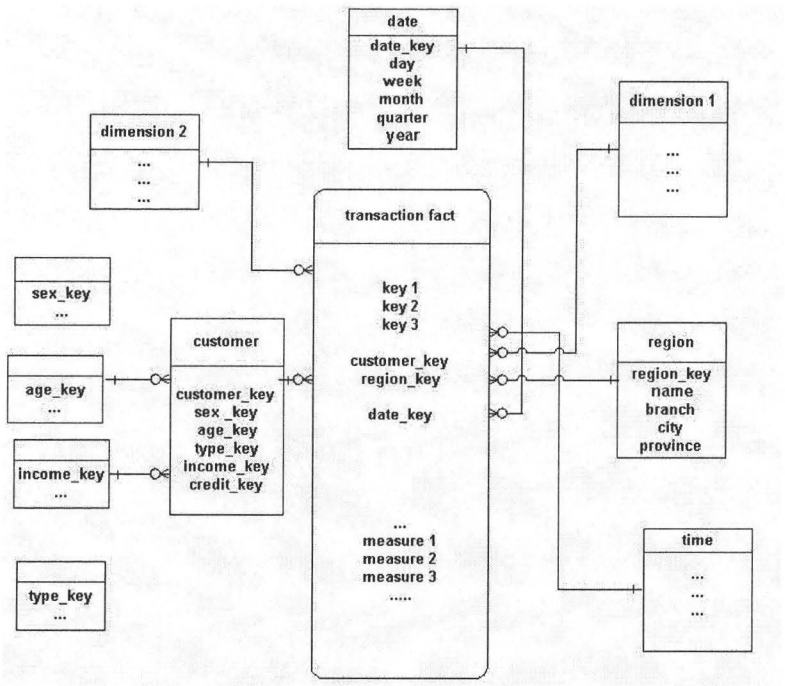


Figure 6.15: Model 3 - Reflect Dimension Category/Level in Dimension Table

Model 4

We classify the group of dimensions in the fact table. We can define one dimension table for each dimension category, such as customer age, customer gender, or customer type. These dimension tables connect to the fact table by surrogate keys, see Figure 6.16.,

It is very fast to analyze information along the separate dimensions of customers. OLAP's performance could improve a lot by this design. However, the fact table numbers of attributes and records increase. More dimension tables also need to be created. If every analytical function required several customer dimensions, this model performance would not be as good since more tables would need to be joined for the

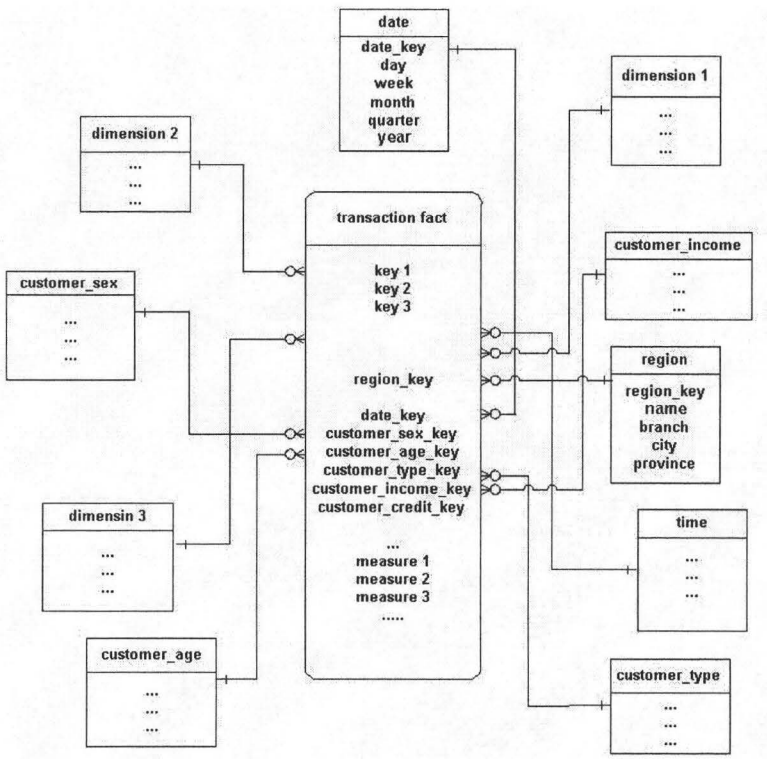


Figure 6.16: Model 4 - Reflect Dimension Category/Level in Fact Table

complete information.

As we can see, an analytical requirement could be satisfied using several data models. They are suitable for different data volumes and business requirements, respectively. When we design a data model, we need to find the best one matching the data volume, system performance and other conditions together. There is no data model suitable for all applications.

6.3.5 Star Schema of Transaction Analysis Model

In S Bank data warehouse project, we basically followed the star schema to design S Bank's data warehouse. Due to the volume of data, we tried to design a model with less data redundancy and kept the fact table size within reasonable limits. We decided do not adopt a snow flake schema or a multidimensional structure, for it may cause worsening of performance.

In the following, we will use a currency trading transaction analysis model as an example and demonstrate how we designed a data model for analytical functions from the ER model for operational system.

A part of FCTS transaction ER model is shown in Figure 6.17:

From the ER model and given the business analytical objectives, we first define seven views of the currency transaction: region, date, time, transaction source, money combinations, money no., and customer. We analyze the transactions from those dimensions. The next step is to define the appropriate attribute and grain for each dimension. After all dimension's elements are defined, we need to identify what measurements need to be analyzed and create the fact table. This leads to the star schema given in Firgure 6.18.

Dimension Table

Dimension tables and supporting dimension objects are defined by what attributes the end users can analyze with their data. Some information are already included or could be derived from the `hist_log` table such as `date`, `time` and `transaction source` while some other information such as `money_combo`, `money_type`, `region`, and `customer` need to be retrieved from related tables. The more detailed customer

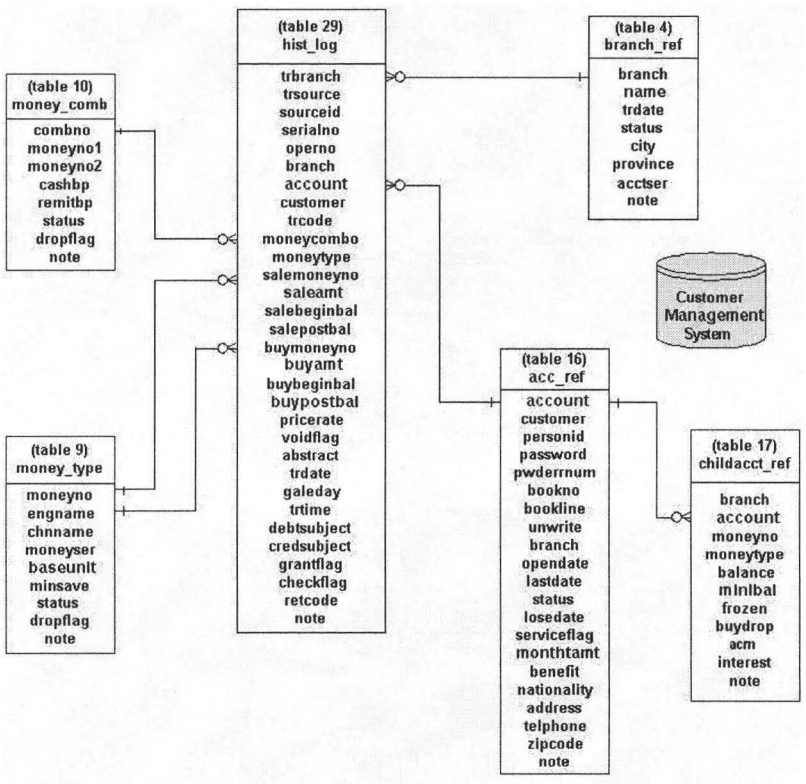


Figure 6.17: FCTS Transaction ER Model

information could be achieved from CMS.

The following dimensions will be created:

1. `Money_no`

   There are 11 currencies in the dimensional table: USD, JPY, DEM, GBP, HKD, CHF, FRF, CAD, AUD, NLG, EUR. `Money_no` is a static dimension and 11 records will be inserted into this dimensional table when the system is initialized.

2. `Money_combo`

   S Bank allows 55 combinations based on 11 different currencies. It is also a

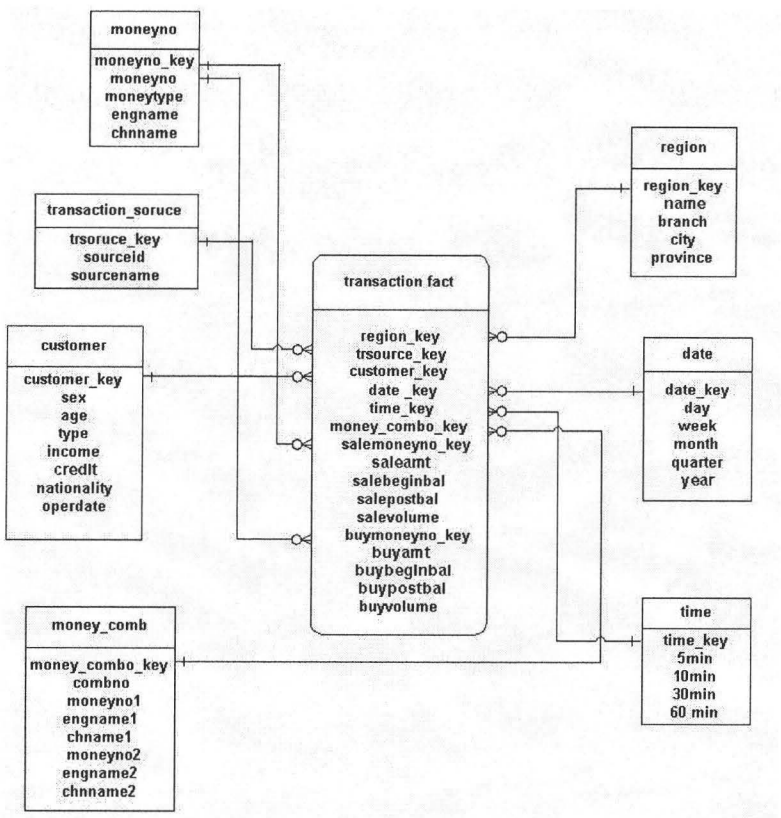


Figure 6.18: Star Schema of Transaction — FCTS data warehouse

static dimension.

3. Transaction source

   All transactions are from the internet, telephone, teller, or self-terminal. The information could be derived from transaction source id in `hist_log` table. It is a static dimension.

4. Date

   There are four levels: year, quarter, month, and day. One can drill up/down the relationships between the various dimension attributes (e.g. quarter drills

down to month and month drills down to day).

5. Time

   According to business requirements, the level of time will be set as 5 min, 10 min, 30 min and 60 min.

6. Region

   For the analytical purpose, we determine 3 levels: province, city and branch.

7. Customer

   This dimension is very complex. The data from different tables and an external system provides the detailed personal information.

Fact table

Fact tables and supporting summaries will define and store the business metrics that the end-users would like to analyze (e.g. transaction volumes, customer transaction amount).

1. Granularity

   Basically, we set granularity level according to the business requirements. However, granularity is the key factor to decide the size of database and query performance. In some cases, we have to consult with users and try to set the granularity more reasonably. This will facilitate validation and user confidence, and maximize the flexibility of future analysis.

2. Partitioning

   Most database vendors provide solutions to partition database. Since the fact

tables are large, each fact table should be partitioned to improve access performance. In our system, most tables range is partitioned by region and date surrogate key (one partition per month). This is a similar partitioning scheme to the future fact tables, which can be used as a model for the new fact.

3. Fields

   Surrogate keys of all dimensions will, of course, be present. It must be invisible to the reporting layer, specifically in order to enhance verifiability of analysis results.

This star schema design we proposed provides information to analyze currency trading transactions from several views. S Bank can get the information about what kinds of customer would like to invest money in currency market, e.g., their professions, ages, and income levels. According to the information, S Bank could adjust the benefit point for a trade quote for a certain amount of transaction volume and attract the customers to do more transactions using S Bank's foreign currency trading system. From the money view, analyst can forecast the future currency trading trend.

Analytical system not only helps business, but also helps improve the IT systems. For instance, the analysis of the data in the data warehouse revealed that in average about 2/3 of all transactions were conducted using internet or telephone. Often, the transaction flow reached 85% of the system capacity from these two sources alone. It was necessary to upgrade the e-business and telephone banking system before customers would experience difficulties or slowdowns. The S Bank's management approved the funding for upgrading the e-business system because the report from data warehouse system provided clear and strong evidence regarding this issue.

6.3.6 Quote Data Model and Analytical Output

Provided below is a second example for quote analysis model and the analysis output based on data from the data warehouse system.

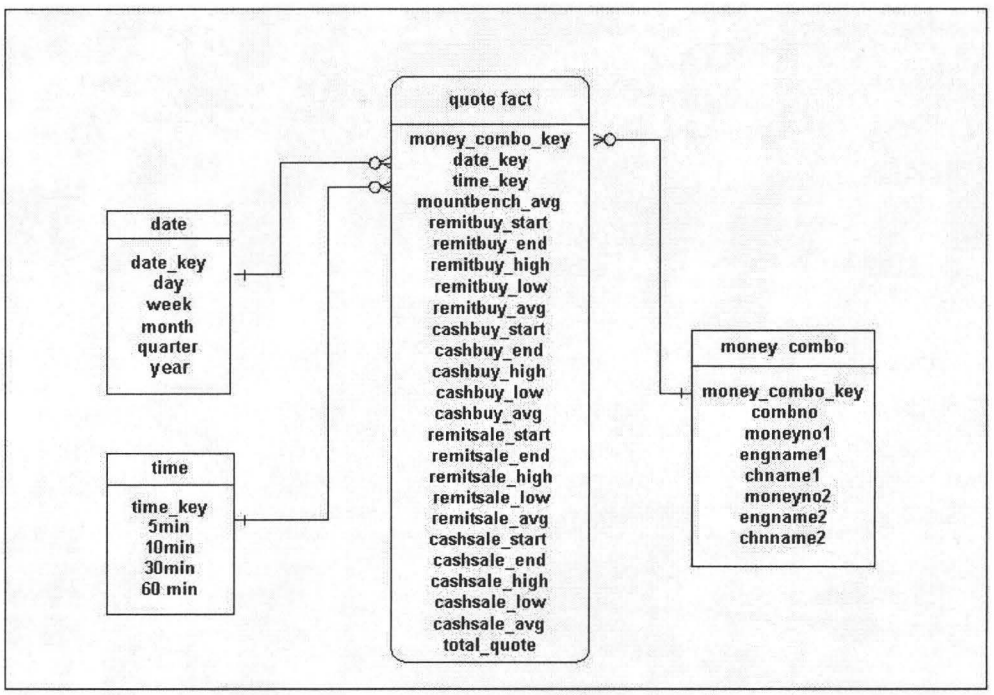


Figure 6.19: Star Schema of Quote — FCTS data warehouse

We still use `his_quote` for this example. The measurements for analysis are decided by business department. Also, they are required to generate analytical results for different currency combinations. There are two time scales: time and date. The granularity of date and time dimension is created according to business requests. Because of the special characteristics of the trading system, we need some analytical results detailed in 5-min, 10-min, 30-min, and 60-min units. Day, week, and month

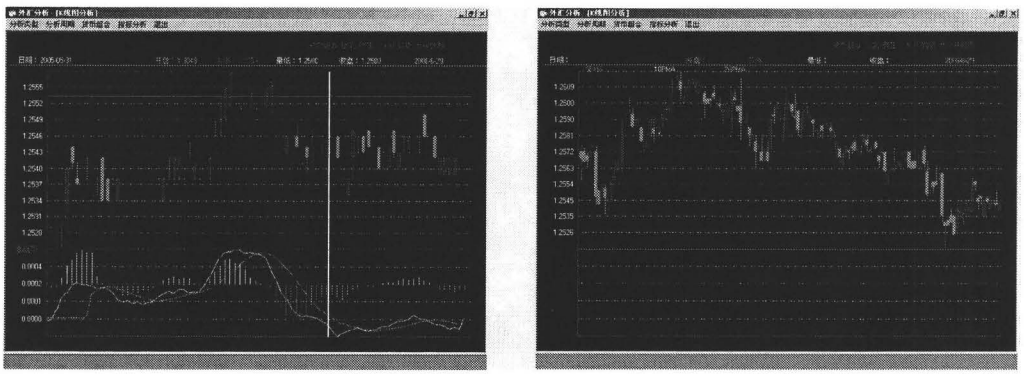


Figure 6.20: 5-min and 60-min Quote Charts

information are also required. In order to make aggregation easier, we separated the day and time dimensions.

The star schema design for the table `his_quote` is given in Figure 6.19. Based on the analytical function, a lot of attributes were added into the fact table. Different from operational data which captures a certain individual record, these fact measurements are about a group unit, such as the average quote in 5 or 10 minute intervals. At first, business department wanted to set the grain at 1 minute level. However, we concluded that this granularity created way too much data, since often we could get more than 5 patches of quote in just 1 minute. For a long-term analysis, such a small unit would not help too much. As we mentioned before, the data granularity directly affects the future data size. After explaining the situation to the management of the business department, they revised the requirements. They decided to capture the analysis quotes for every 5, 10, ... 60 minutes. These levels are created in the time dimension. Day, week, month, quarter, and year level are setup in the date dimension.

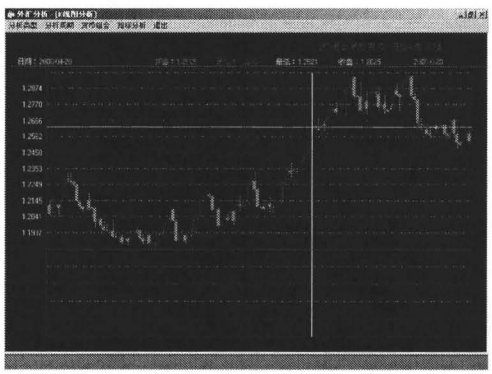


Figure 6.21: Daily Quote Chart

The end-user layer generates reports, charts, or other forms of analytical results based on aggregate data provided by FCTS data warehouse system. For illustration, we give examples of 5-min and 60-min charts for the EUR/USD quote analysis in Figure 6.20.

The daily quote chart is given in Figure 6.21.

6.3.7 Conformed Dimension in Data Model

Some dimensions actually could be shared by both fact tables. A conformed dimension is a dimension that is shared by two or more fact tables.

The advantages of conformed dimensions include:

1. help design a concise data model;
2. save time and efforts because several subjects share one dimension table;
3. can help join fact tables easily; and
4. can maintain data consistency more easily.

In the real world, there are always new analytical objectives from the business

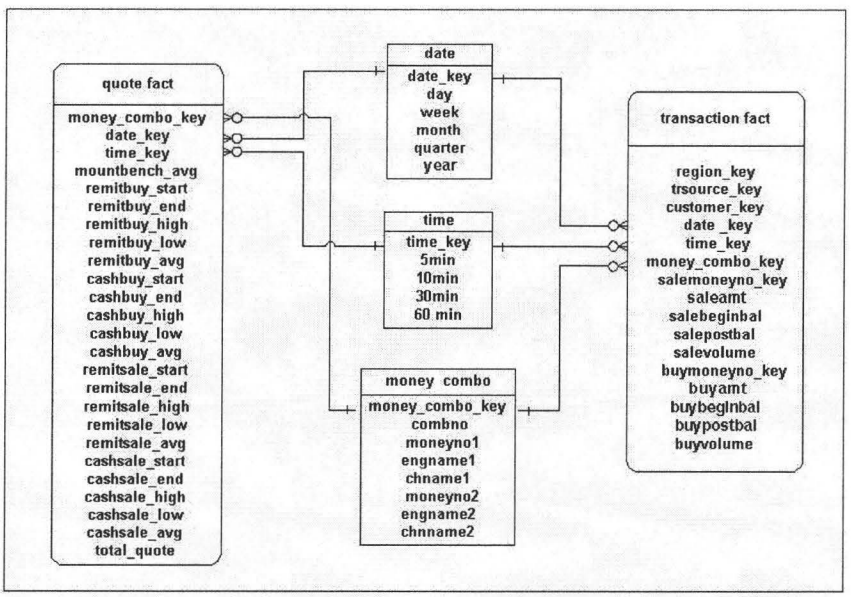


Figure 6.22: Conformed Dimension Tables

department, but the experience tells us that the views for the new analytical objectives are often similar to the previous ones. They will not change as frequently as analytical facts. In FCTS data warehouse system, there are several conformed dimension tables shared by different facts.

Chapter 7

Conclusion

In this thesis, we briefly described the concepts, methods, and important components of a data warehouse. The description is intended for students taking an undergraduate database course. In order to understand the data warehouse theory, we illustrated some details of design and implementation of data warehouses on a case study of data warehouse design for S Bank. We also compared different data models for various situations and two system designs for S Bank FCTS data mart project. Most people learn a lot from involvement in a data warehouse project. It is a really valuable experience to participate in such a project. Some knowledge and experience cannot be learned from textbooks. In the rest of this conclusion, we would like to summarize our findings based on the experience.

We started with the definitions and discussed the special characteristics of data warehouse. It is important to understand that a data warehouse is to support business analysis and decision making. The special characteristics of a data warehouse determines that we cannot use the same database for both daily transactions and analytical functions. The final goal leads to a special data model and system design to build a database for the analytical system which is different from traditional

operational database systems.

Although there are many strategies for system design and data model, we found that there was no standardized process when implementing a real world project. When participating in design and implementation of data warehousing system in real world, we face a more complex and more complicated situation than textbook examples illustrate. There are no simple rules to follow. We have to consider various factors, such as funding, people, skills, requirements, time, and existing business systems. Some of them really restrict the system design, architecture, and data model that could be used. The ideal and desired design must often be modified so it can be implemented within the reality constraints.

Moreover, the data warehouse system design and architecture still evolve despite the more than 10 years of history. In the early versions, the architecture was simpler; in terms of data sources, data warehouse, and end-user layer. Nowadays, most applications will set a data staging area separated from the data warehouse itself or keep copies of relational data on the data warehouse server.

For enterprise level data warehouse system, traditionally most implementations set up a global data warehouse, and then several data marts fed from the warehouse. However, this extends the development time and leads to inflexible business analytical function. If departments do not share too much common data, it is much easier and faster to set up several data marts fed from data staging area(s) directly.

Recently, I discussed these issues with a designer of a data warehousing system for Shoppers' Drugmart™ company. They used a similar architecture we used in S Bank and described in this thesis. In their design they uses individual records as the smallest granularity for data marts. This approach differs substantially from

the typical concept presented in most data warehouse textbooks stipulating that data warehouse is suitable only for aggregate data and and does not keep individual records. However, this kind of design will satisfy the business requirements of drilling down to operational level of records within the data warehousing system.

Most corporate data mart sizes range from small to medium. Thus, they are easy to manage and maintain. As we mention many times in this thesis, the granularity is the most important design aspect which impacts the database size and data access performance. In general, if end-users are clear of their needs from a data warehouse and the long term business objective, a larger granularity can be determined. Otherwise, we are forced to keep the original operational data in the data warehouse server in case there are new requirements which go beyond the original assumption of granularity. The operational data could be stored in the same physical server, or the data staging area, or the presentation server.

In my experience, the dimensional data model is more suitable and easier for data warehouse system design and development. After understanding the business requirements, we identify the dimension and the fact tables which constitute the star schemas. A star schema represents different views to analyze a particular object.

Both ROLAP and MOLAP are based on multidimensional view of the analytical data. MOLAP has good query performance since there are physical cubes providing pre-calculated data once the end-user poses a query. Unlike MOLAP, there are only virtual cubes in ROLAP. SQL executables are able to transfer an OLAP query to a multidimensional data view for the end-user. Since end-users only face the multidimensional data views, they may not know which approach is used to organize and store data in data warehouse.

Within the past few years, vendors developed and are offering powerful and efficient data warehouse and client OLAP tools. Most vendors thus provide a whole system solution, from the database, the ETL tools, to the end-user layer analytical tools. When designing a data warehousing system, we have to consider the final analytical software and data mining tools, since data warehouse is to provide a data foundation for them. The compatibility and performance between a data warehouse and a client are critical factors for a successful data warehouse system. Oracle™, IBM™, SAS™, NCR™, and Essbase™represent the top layer of companies offering data warehouse systems and related software. Most of them provide their own client analytical tools or their products are compatible with other companies software. Relational database products have been optimized for OLAP function recently, including parallel storage, parallel query, and data management that sharply improve data retrieval performance. These new features and enhancements strongly support ROLAP applications.

Bibliography

[1] IDC Organization, *A Study of the Financial Impact of Data Warehousing*, IDC, 1996

[2] B.J. Haley, *Implementing the decision support infrastructure*, University of Georgia, 1997

[3] Inmon, W. H. and R.D. Hackathorn, *Using the Data Warehouse*, New York, John Wiley & Sons, 1994

[4] Kimball, R., *The data warehouse Toolkit*, John Wiley & Sons, 1996

[5] Mattison, R., *Data Warehousing: Strategies, Technologies and Techniques*, McGraw-Hill, 1996

[6] Chamoni, P., Gluchowski, P. (Eds). *Analytische Informationssysteme: Data Warehouse, On-Line Analytical Processing, Data Mining*, Berlin, Springer-Verlag, 1999

[7] `http://www.cs.usask.ca/resources/tutorials/csconcepts/2000_12/Tutorial/Lesson1/Lesson1.html`

[8] Inmon, W. H., *Building the data warehouse*, Third Edition, John Wiley & Sons, 2005.

[9] Silberschatz, A., H.F. Korth, and S. Sudarshan, *Database System Concepts*, Fifth Edition, McGraw-Hill, 2006

[10] Kimball, R., *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*, John Wiley & Sons, 1998.

[11] Codd, E.F., S.B. Codd, and C.T. Salley *Providing OLAP to user-analysts: an IT mandate*, Technical Report, E.F. Codd and Associates, 1993, also available at URL
`http://dev.hyperion.com/resource_library/white_papers`

[12] `http://www.o2olap.com/Business%20Solutions/OLAP/OLAP.aspx`

[13] `http://bloxchronicles.blogspot.com/2006/04/rolap-molap-holap-for-beginners.html`

[14] *The case of Relational OLAP*, MicroStrategy White Paper, 1995

Other reference material

[15] Hobbs, L., S. Hillson, and S. Lawande *Oracle 9iR2 Data Warehousing*, Elsevier Science, 2003.

[16] *SAS 9.1.3 OLAP Server: Administrator's Guide*, SAS Publishing, 2004.

[17] Lechtenbörger, J., *Data Warehouse Schema Design*, Dissertations in Database and Information Systems, Volume 79, 2001
also available at
`oesen0.informatik.uni-leipzig.de/proceedings/paper/diss2.pdf`

[18] Filipe, J., B. Sharp, and P. Miranda, *Enterprise Information Systems III*, Kluwer Academic Publsihers, 2002

[19] Devlin, B., *Data Warehouse from Architecture to Implementation*, Addision-Wesley, 1997

[20] Chamoni, P., Gluchowski, P., *Alternative data warehouse architectures*, PUBLISHER, 1999.

[21] Chi Taiwen, *Data warehouse design and implementation*, PUBLISHER, 2005.

[22] Lin Yue and Guo Lingyun, *Data warehouse theory and application*, PUBLISHER, 2003.

[23] Yue Zhongming and Liu Yiyu, *Data warehouse project management*, PUBLISHER, 2006.

Abbreviations

AUD	Australian Dollar
CAD	Canadian Dollar
CHF	Swiss Franc
CMS	Customer Management System
DEM	Deutsche Mark
DBA	Database Administrator
DSS	Decision Support System
EIS	Executive Information System
ETL	Extract, Transform and Load
EUR	Euro
FCTS	Foreign Currency Trading System
FRF	French Franc
GBP	British Pound
HOLAP	Hybrid Online Analytical Processing
HKD	Hon-Kong Dollar
JPY	Japanese Yen
MDDB	Multi Dimensional Database
NLG	Dutch Guilder
MOLAP	Multidimensional Online Analytical Processing
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
RDB	Relational Database
ROLAP	Relational Online Analytical Processing
SQL	Sequential Query Language
USD	US Dollar