Design Issues in Data Warehousing: A Case Study

Kan Yao

A Thesis
In
The Department
Of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

March 2003

ABSTRACT

Design Issues in Data Warehousing: A Case Study

Kan Yao

Data warehousing is the foundation of DSS (decision support system), of which the goal is to enable decision makers to make better business decisions based on analysis of historic data related to the business operation. Data warehousing has become a major business trend, both for product and service sectors and for application to daily business in all industries.

In this thesis, we compare and discuss the two data models and two methods of OLAP (MOLAP and ROLAP). The two data models and two OLAPs are not mutually exclusive in any project. An example of the hybrid design is presented in a case study. In this project, we are able to apply two data models at different stages of the data warehouse. By combining ER model, Star Schema and aggregate table in one architecture, we are able to maximize on both storage capacity and query performance. In addition to the selection of data models, the choice between MOLAP and ROLAP is discussed. When the objectives of the client are clearly defined, the functional requirements of the data warehouse are determined, and the amount of data is relatively small, MOLAP would likely be a better choice because of its superior performance and intuitive querying mechanism. ROLAP becomes a more suitable candidate for OLAP when the amount of data is exceedingly large, and the client does not want to leave out anything. Again, it is possible, even desirable, for both OLAP methods to be applied in one project to compliment each other.

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to Professors B. C. Desai for his guidance and assistance in the preparation of this manuscript.

Special thanks to Dr. Shiping Liu, founder and CEO of GBICC, for giving me the opportunity to work on the project and for helping with the completion of this manuscript. I would also like to thank all the colleagues in GBICC for their valuable input.

Last but not the least, thanks to my parents and my husband. They make everything in my life possible. My gratitude is simply beyond words.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Data warehousing is the foundation of DSS (decision support system), of which the goal is to enable decision makers to make better business decisions based on analysis of historic data related to the business operation. Data warehousing has become a major business trend, both for product and service sectors and for application to daily business in all industries. According to the *META Group*, the data warehousing market, including hardware, database software, and tools, had grown from $2 billion in 1995 to $8 billion in 1998[42]. Today, data warehousing is still one of the most talked-about business technologies in the corporate world. The market survey for data warehousing solution pointed out that the worldwide data warehousing market will expand at a compounded annual growth rate of 43%, reaching $148 billion by 2003. The US market alone will account for $72.7 billion of the data warehousing market share by 2003, while growing at 41% annually [43]. Industry appears mostly concerned with multidimensional modeling of data warehouse [3, 4, 11, 21, 23], multidimensional DBs [3, 23, 22], Online Analytical Processing (OLAP) [3, 4, 11, 18], and Relational OLAP [11, 24, 25].

A data warehouse is a "subject-oriented, integrated, time-varying, non-volatile collection of data that is used primarily in organizational decision making." [2] In general practices, the data warehouse is a separate hardware and software installation based on the company's operational databases. The main task of data warehouse is to support on-line analytical processing (OLAP) and data mining. The functional and performance requirements of OLAP are quite different from those of the on-line transaction processing (OLTP) applications traditionally supported

by the operational databases [29]. Online transaction processing (OLTP) application performs clerical data processing tasks such as order entry and banking transactions that are essential to the daily operations of business entity. Data warehouses, in contrast, are targeted for decision support. In data warehousing, historical, aggregated data is more important than detailed individual records.

Data warehouses might be implemented on standard or extended relational Database Management Systems (DBMS), called Relational OLAP (ROLAP) servers. These servers assume that data is stored in relational databases, and they support extensions to SQL and special access and implementation methods to efficiently implement the multidimensional data model and operations. In contrast, multidimensional OLAP (MOLAP) servers directly store multidimensional data in special data structures (e.g., arrays) and implement the OLAP operations over these data structures.

Despite the abundance of commercial offerings of turn-key solution, there is more to building and maintaining a data warehouse than the mere selection of an OLAP server and defining star schema and some complex queries for the warehouse. Organizations want to implement integrated enterprise data warehouse that are tailor made to their own business nature and organization. However, designing and implementing an enterprise warehouse is by no means a simple process. It not only requires extensive business rules analysis, but also careful design of architecture which includes selection of correct hardware and software tools.

In this thesis, we discuss in general terms the design of a data warehouse. Then we will present a data warehouse design for actual business as a case study. The thesis is organized as the follows: Chapter 2 presents a brief history of data warehouse and its evolution. Chapter 3

discusses the conceptual design of a data warehouse and its main components. Chapter 4 discusses data modeling and OLAP. Chapter 5 presents a case study in which we present the design of a data warehouse project. Chapter 6 is a brief summary and conclusion to our study.

Through the discussions in the following chapters, we hope to present a fair picture of the advantages and shortcomings of two data models: relational data model and multidimensional data model. Different from many existing publications on this subject, it is not our goal to support one model over the other. Instead, we will show that the two models can coexist in a data warehouse, complimenting each other.

In addition to data models, it is also important to discuss the two types of OLAP servers. Again, the industry and academic have divided opinions about which one of the OLAP servers is superior to the other. It is our belief that either one of the OLAP technology does not have clear advantage over the other one in any given case. Rather, the choice between ROLAP and MOLAP in any project should be based on the characteristics of the business nature and the data warehouse itself.

Chapter 2

Data Warehouse Design

In this chapter, we will start with the basic concept and major elements of the data warehouse, followed by some important processes involved as the data flow from the sources down to the hands of the users.

Most of the definitions in this chapter are cited directly from Ralph Kimball's book, "*The Data Warehouse Lifecycle Toolkit*", since there are no universal names and definitions on terms used in designing and/or maintaining a data warehouse and the terms given in this book are "as close to mainstream practice as they can make" [4].

2.1 Concept of Data Warehouse

A data warehouse is a subject oriented, integrated, non-volatile, and time variant collection of data in support of management's decisions. [2].

- "Subject-oriented" means the data are organized by business topic, not by customer number or some other key.
- "Integrated" means the data are stored as a single unit, not as a collection of files that may have different structures or organizations.
- "Non-volatile" means that the data don't keep changing. New data may be added on a scheduled basis, but old data aren't discarded.
- "Time-variant" means that a time dimension is explicitly included in the data so that trends that aren't changes over time can be studied. This does not mean that data

elements change over time, as a checking account balance would in a bank's operational database.

Some other definitions from other authoritative sources:

A single, complete and consistent store of data obtained from a variety of sources and made available to end users in a way they can understand and use in a business context [1].

The data warehouse provides access to corporate or organizational data; the data in it is consistent; the data in it can be separated and combined by means of every possible measure in the business; it is also a set of tools to query analyze, and present information; it is the place where we publish used data; the quality of the data in it is a driver of business reengineering [3].

These definitions are quite different on the surface. Yet several common threads emerge from reading them as a group:

a. A data warehouse contains a lot of data.
b. The data warehouse is organized so as to facilitate the use of this data for decision-making purposes.
c. The data warehouse provides tools by which end users can access this data.

The major elements of data warehouse, and the major external entities with which a data warehouse interacts, include:

a. The transaction or other operational database(s) from which the data warehouse is populated. External data is also fed into some data warehouse. A key point here to be clear is that data warehouse only gets a copy of the transaction data. It does not

store transaction data directly.

b. A process to **extract** data from database(s), and bring it into the data warehouse. This process must often transform the data into the database structure and internal formats of the data warehouse.

c. A process to **cleanse** the data, to make sure it is of sufficient quality for the decision making purposes for which it will be used.

d. A process to **load** the cleansed data into the data warehouse database. The four processes from extraction through loading are often referred to collectively as **data staging.**

e. A process to create any desired **summaries** of the data: pre-calculated totals, averages, and the like, which are expected to be requested often. These are stored in the data warehouse along with the data imported from internal and external sources.

f. **Metadata**, "data about data." It is useful to have a central information repository to tell users what's in the data warehouse, where it came from, who is in charge of it, and more. The metadata can also tell query tools what's in the data warehouse, where to find it, who is authorized to access it, and what summaries have been pre-calculated.

g. The **data warehouse database** itself. This database contains the detailed and summary data of the data warehouse. Metadata can be placed inside or outside database. It is a simple matter of preference. Because the data warehouse isn't used for processing individual transactions, its database doesn't have to be organized for transaction access and retrieval patterns. Instead, it is organized for different access patterns used for analysis.

h. Query tools. These usually include an end-user interface for posing questions to the

database, in a process called **online analytical processing (OLAP)**. They may also include automated tools for uncovering patterns in the data, often referred to as **data mining**. A give data warehouse must have at least one of these two types and may have both

i. The user or users for whom the data warehouse exists and without whom it would be useless.

2.2 Design Issues

2.2.1 Granularity

The single most important aspect of design of a data warehouse is the issue of granularity. Granularity refers to the level of detail or summarization held in the units of data in the data warehouse [2]. In another word, the more detail there is, the lower the level of granularity. The less detail there is, the higher the level of granularity.

The granularity largely affects the volume of data that resides in the data warehouse and at the same time affects the type of query that can be answered [30]. The volume of data in a warehouse is traded off against the level of detail of a query. The strength and weakness of both cases can be shown by following table:

Table 2.1: Characteristics of Granularities:

	Low Granularity	High Granularity
Strength	- able to answer any kinds of queries	- more efficient way of representing data
Weakness	- takes a lot of resources to locate a record	- unable to definitively answer detailed queries

It is obviously a space problem. But in real world, it is rare that only a single event is examined. It is much more common that a collective view of data is taken. This means that a large number of records would be examined frequently than a single record. This behavior is in favorable of higher granularity.

Most of the time, there is a great need for both efficiency in storing data and its access and for the ability to analyze data in great detail. Then, dual levels of granularity are introduced.

This type of granularity fits the needs of most organizations. Actually, because of the costs, efficiencies, ease of access, and ability to answer any query that can be answered, the dual level of data is the best architectural choice for the detailed level of the data warehouse for most shops.

There are two types of data in this kind of data warehouse, "true archival" detail data and lightly summarized data.

- At the true archival level of data, all the detail coming from the operational environment is stored. Since the volume of data is large, it is reasonable to store it on a medium such as magnetic tape. Answers to queries of greater details will be delivered from here.
- The volume of lightly summarized database is significantly less than the detailed database since it is compact and efficiently accessed. And there will be a limit to the level of detail that can be accessed in this database. Most decision support system (DSS) processing goes against lightly summarized data.

2.2.2 Partitioning

Partitioning of data refers to the breakup of data into separate physical units that can be handled independently. Data is partitioned when data of a like structure is divided into more than one physical unit of data [2]. The issue here is not whether partitioning is needed, but how it should be done.

Small physical units of data provide much more flexibility in the management of data than do large physical units. When data resides in large physical units, among other things it cannot be [2]:

- Restructured easily
- Indexed freely
- Sequentially scanned, if needed
- Reorganized easily
- Recovered easily
- Monitored easily

It is developer's choices for partitioning data by following certain criteria. But, it is almost mandatory that one of the criteria for partitioning be by date in a data warehouse environment. Some of commonly used criteria are:

- By date
- By line of business
- By geography
- By organizational unit
- By all of the above

Another decision in partitioning data to be made by the developer is whether to do partitioning at the system level in which it is a function of the DBMS and the operating system, or at the application level in which it is done by application code and is strictly under control of developer and programmers. It is almost becoming a rule to set partitioning at application level due to its flexibility of management [30].

Finally, according to W.H.Inmon, the acid test for partitioning of data is to ask the question "Can an index be added to a partition with no discernible interruption to other operations?" [2] A positive answer indicates the partition is fine enough. Otherwise, more work will be needed.

2.3 Elements of a Data Warehouse

A typical data warehouse is constructed with three parts based on their functionalities. Figure2.1 shows a rough architecture of a data warehouse.

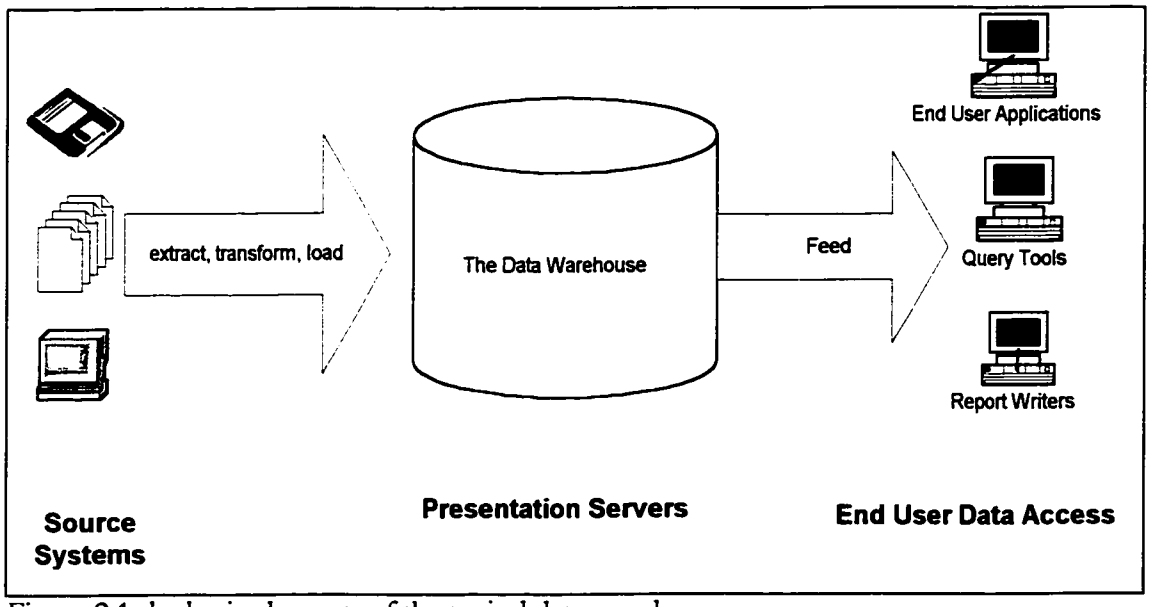

Figure 2.1 the basic elements of the typical data warehouse.

The source system contains all the data which will be re-directed into the data warehouse. It keeps the most detailed daily transactions of the business. The data warehouse itself, however, does not store the transaction data directly. The source system is not queried as broad and unexpected as data warehouse. Presentation server is the central place where data is stored and presented. It receives cleaned and integrated data from the source system, and provides support for the end users by answering their queries, yielding information for reports and allowing accesses of some other applications. Finally, the end user data access is the "front room" of a data warehouse system. It is the part responsible for delivering data to the user community [4]. It is able to provide services such as warehouse browsing, data access and security, query management and standard reporting etc.

2.3.1 Source System

The Source System, sometimes called "the Legacy System", is formed by the transaction or other operational database(s) from which a data warehouse is populated. Quite frequently, external data is fed into some data warehouse [9]. Unlike querying in data warehouse itself, queries to the source system are more towards current events, much more focused, "account-based" and restricted. Typical queries here could be "What is the current balance by account number 0123456789?" or "List all transactions made in last 7 days of account 0123456789" etc. It is clear that each record in the source systems has a unique system key to separate it from other records. These keys are called *product keys* [4], and they are saved as attributes in the data warehouse. Again, the transaction information stored in the source system will NOT be

loaded directly into the data warehouse. Complicated data acquisition processes will be applied to the data before they can enter the data warehouse.

Another important characteristic of the source systems is that they maintain little historical data and data consistency is low. Using a banking transaction list as an example, client usually receives a monthly account statement which lists every single transaction, such as the time the transaction was performed, and the amount the transaction involved. However, it is impossible for a client to go into the bank and obtain the record of a transaction he or she made years ago. The bank usually takes a few days to weeks to process the request before returning the results to the client. The time it takes to process such request is mainly the result of little historical data saved in its source system. In the sense of data consistency, using a date field as an example, the date field used in the same bank but among different geographical locations could be very different. Some branches opened 10 years ago could use date format as "yyyy-mm-dd", other newly opened branches could define the date format as "mm/dd/yyyy". Also, the source systems have direct contacts with human beings, who simply can make careless mistakes.

Furthermore, data storage types are system dependent. Information Management System (IMS), Integrated Database Management System (IDMS), relational database from IBM (DB2) and flat files are all very common. However, flat file is one of the standard source files for the data warehouse.

Finally, the data models of the source systems are not unique in different source environment. Different operational databases may be constructed during different time period and/or in

different geographical locations designed by different project groups. The data model therefore may vary from hierarchical model, fully denormalized flat files to third normal form.

2.3.2 Presentation Servers

Presentation Servers are the target platforms where the data is stored for direct querying by end users, reporting systems and other applications [4]. But before any further discussion on the servers is presented, another important element – data staging area, which is starting to gain more attentions currently and can be found in more and more data warehouse architecture design, will be discussed below. Presentation Server will be discussed in the subsequent section.

The Data Staging Area

Data staging area is "everything" in between the source system and the presentation server. Everything here means completely everything, every process and every single bit of the storage. Formally, it is "a storage area and set of processes that clean, transform, combine, de-duplicate, household, archive, and prepare source data for use in the data warehouse" [4]. In short, it is a construction site for the data warehouse. The staging area is not intended to be seen by the users of the data warehouse. A staging area is the key to providing data quality and integration in a data warehousing environment. [8] Data from source systems is first extracted and integrated into the staging area where its quality is checked and maintained. Clean and consistent staging area data is then used to populate the various information stores of the underlying data warehouses and data marts that make up the data warehousing system.

Some data warehouse projects do not employ staging areas. Although there are situations where staging areas are not required, their use provides significant benefits as data warehouse projects grow in number, especially if a significant amount of legacy source data is involved, such as in financial organizations. Figure 2.2 shows a typical data warehouse equipped with a separate stand along data staging area.

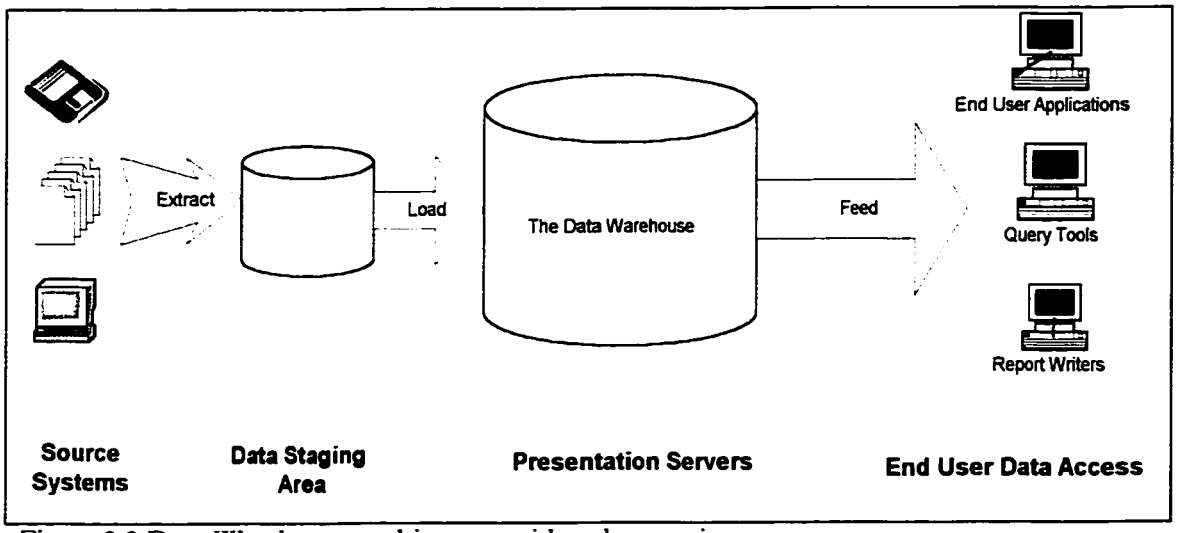

Figure 2.2 Data Warehouse architecture with a data staging area.

The reasons for having a solid staging area are:

- Staging area is the best place for emergency data recovery. Staging area can be designed to perform certain level of archiving and storage. Cleaned and transformed data may be saved for a certain period of time. During this time, if anything happens, this is the place to find the most recent data in correct form.
- For a multidimensional data warehouse, dimensions can be created at this stage and they can be simply replicated into the warehouse.

The overall structure of the staging area can vary from one single centralized solid piece to a wide spread network over a large number of machines.

Although source system, data staging area are both important components of a functional data warehouse, source area is usually considered to be outside of the data warehouse since no direct manipulations can be added to the contents and the format there. At this stage, all data stored on the presentation servers had passed through the necessary cleaning and transforming processes successfully, and are waiting to be queried by the end users and/or various applications.

Three types of data are stored on the presentation servers, raw data, aggregated business process data and metadata.

- Raw data is simply record keeping. It holds "data at the lowest-common-denominator level, that is, at the lowest level of detail necessary to meet most of the high-value business requirements" [4]. Raw data is sometimes called "atomic data" since it consists of individual data items. It is the lowest level of data stored in the presentation server. All functions provided by the data warehouse use directly the raw data or data derived from the raw data. In some cases, raw data can be understood as transaction data plus the time dimension. Examples of raw data can be:

  Mr. Wang's checking account balance on Jan. 1st, 2002 was $3603.56.

  Mr. Wang's checking account balance on Feb. 1st, 2002 was $3991.23.

  ……..

- Aggregated business process data is chosen and/or derived from raw data to serve specific business needs. Consequently, aggregated business data is time-dependent and requirement-based. It is mostly targeting for business analysis purpose and to speed up the performance of common queries by keeping most frequently queried data and

those contained summarized information. These data can be calculated ahead of time and stored in the data warehouse database for later usage. Examples are:

By Feb. 28th, 2000, total of 356,264 clients bought RSP in branch number 23.

By Feb. 28th, 2001, total of 297,126 clients bought RSP in branch number 23.

.....

- Metadata can simply be understood as "data about data". Metadata are data that describe the data in a data warehouse [9]. It serves many proposes from mapping data sources to information within a data warehouse during extraction and load processes, automating the production of summary tables during warehouse management process, to directing query to the most appropriate data source during a query process. [5]

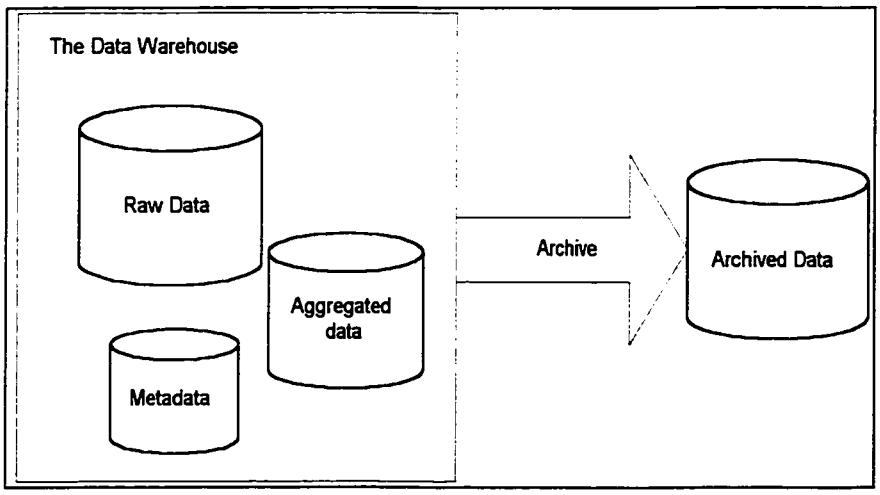

Figure 2.3 Contents of "The Data Warehouse"

All these data will be periodically archived or cleaned according to the requirements of the business. Figure 2.3 shows the contents of the data warehouse and its archiving. Since this report is focused on the OLAP application, the discussion of metadata is out of the scope of this report.

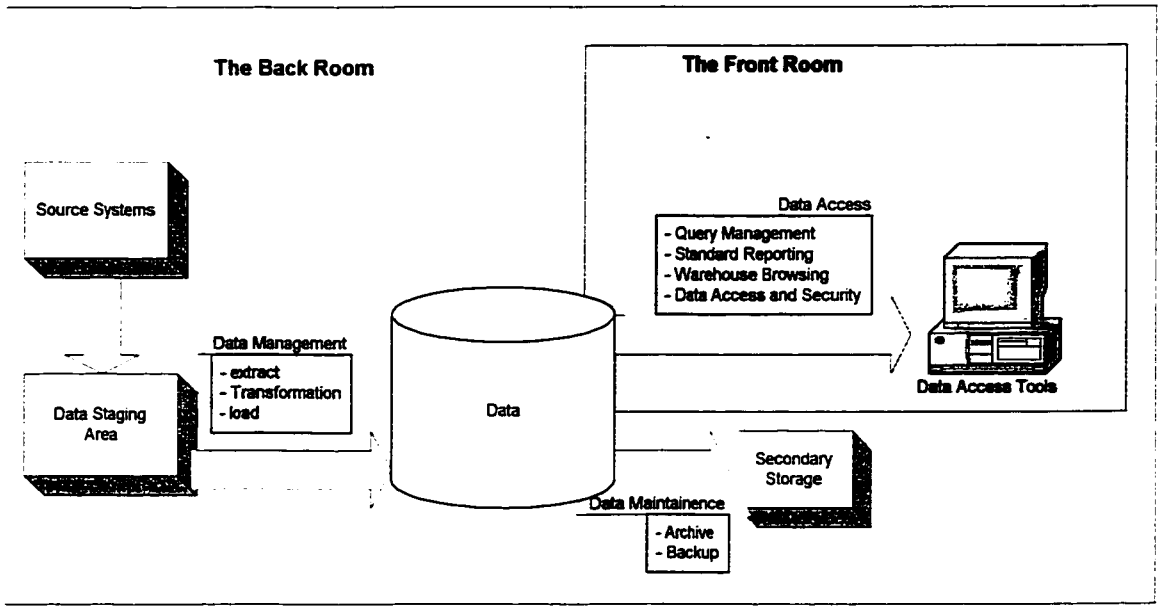

Figure 2.4 Technical Architecture

All the systems discussed so far are sometimes understood as "back room architecture". Back room can be seen as the engine room of the data warehouse. Figure 2.4 shows a technical architecture of the whole data warehouse. The backroom actually includes the source systems, data staging areas, data management processes which is usually called ETL (Extract, Transformation, Load) processes, data warehouse central database and the secondary storage for archiving and backing up the out-of-date information. The front room, the public face of the warehouse, will be discussed in more detail in the following section.

2.3.3 End User Data Accesses

The end user data accesses are a complete set of tools and the services supporting the tools. These tools maintain a session with the presentation server, sending SQL requests to the server in order to present a report, a graph, or some other higher form of analysis to the end users.

The primary purpose of the data access layer is to hide as much of the complexities of the data system as possible [4]. The major services at this layer are listed according to their importance in ascending order:

- Warehouse or metadata browsing. This service is provided by tools that use metadata as guide helping the end users finding and accessing the information they need. Once the end users find the information, other more advanced services should be able to link with this tool, such as browsers. However, warehouse browsing is not yet the main focus of the most data warehouse.

- Data Access and Security. This is where we control the connections between end users and data. Authorization and authentification are two major concerns here. The level of security should be decided by the level of sensitivity of the data. Data should be allowed to view or change only by those users who have the correct identification. Any failure in authorization and authentification processes will result in destruction of the warehouse which will in turn lead to destruction of the entire business.

- Query Management. According to [4], query management services are the set of capabilities that manage the exchange between the query formulation, the execution of the query on the database, and the return of the result set to the desktop. Query management service is expected to support various query tools for data analyzing. These tools may include end-user interface for posing questions to the database, in a process called "Online Analytical Processing (OLAP)"; they may also include automated tools for uncovering patterns in the data, often called "Data Mining".

OLAP and data mining are two different approaches to analyzing the content of a data warehouse. OLAP is more active and user guided, and data mining is mostly done automatically. One definition of OLAP, from [10], 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 user". End users of the data warehouse are able to "slice and dice" the content of the data warehouse to answer specific questions by using OLAP software. OLAP approach of analyzing content of data warehouse is often chosen when the end users are clear of what they are looking for. More details about OLAP will be discussed in the next Chapter.

2.4 Data Processes

This section focuses on the data preparation and some important data maintenance processes. These processes are sometimes referred to as ETL (data Extraction, Transformation and Load).

These processes are of great importance. Any mistakes or less consideration can lead to unthinkable failures in a data warehouse project. Besides data preparations, data maintenance is another major issue one should not be careless about. It is a general recognized that it is the most cost and labor consuming part of a data warehouse project.

The next three subsections will present the processes of data preparation, namely ETL processes. Data maintenance issue, i.e: data refreshing and data purging, will be covered in the last two subsections. Although these two processes are considered as separate issues from

data preparation processes, it is felt that they are closely linked to data preparation, and is suggested here to be considered as part of the ETL data processes.

2.4.1 Data Extraction

The primary goal of this process is to select useful data out of the source and put them into data staging area. Since extraction process is not as simple as moving all available data onto the data staging area from source systems, one of the most important concerns here is "what to extract". Extracting data can take a long time and one may have to redo it if data extraction specification is not appropriate. In fact, determination of what to extract is closely related to business objectives.

There are several extraction types one can choose, thought they each serve different purpose. [4]

- Incremental Load. It is the most used extraction type. It can be understood as a periodic load triggered by a time stamp, e.g. a date of transaction or some kind of indicator flag in the source system. Only those data which has changed since the last extraction will be loaded onto data staging area if it is applied. This technique is more efficient in both time and space.
- Transaction Events. This technique usually is applied to a source system which doesn't have a reliable time stamp to trigger the load. Instead, the transaction log file is used to identify the updated records for loading.
- Full Refresh. This is the technique which pulls the complete tables. Because of the nature of its operation, there is size limit to this strategy; therefore, it is not often used. It only becomes necessary when loading tables with significant changes. In this case,

full refresh is preferred to the above two methods which are much more time-consuming to determine the changes in the table. One other application for this strategy is during periodic resynchronization between systems.

Some other concerns during data extraction may include multiple sources, code generation and/or compression/decompression. It is rare for a data warehouse that only one type of source file needs to be extracted. Commercial tools are available in helping data extraction. In order to speed up the process in a scaled project, many tools have adapted the parallel loading technique [35, 36]. It is necessary for the people to make sure the chosen tools will work with the key source system; otherwise, extraction codes generation must be scheduled. Finally, if the source systems are wildly spread and in a significant distance, data compression and decompression processes must be considered in order to make transformation more efficient.

2.4.2 Data Transformation

Data stored in the central data warehouse database is presentable to the end users and valuable to the business. The transformation steps between being extracted from the source systems and being presentable data are called data transformation. Sometimes, this whole process is separated into two related processes, transformation and cleaning [31]. However, data cleaning will be regarded as one of the necessary steps grouped under data transformation in this thesis. Some of the actions which take to transform raw data extracted from the source systems to useable state are: data integration, consistency check, renormalization and de-normalization, data type conversion, data aggregation etc.

- Data Integration. This step involves generating surrogate keys which map keys from one system to another.
- Consistency Check. This is the step, suggested to be performed as early as possible in order to increase flexibility and reduce errors. As we all know the importance of data consistency. This is the step to make sure the data is self-consistent as well as consistent with other records from either same or different sources. Examples of inconsistent data among themselves are: data value equals to "2000-11-32", time value equals to "25:34:12", and nonsensical phone numbers, addresses, counts etc. When data is examined against other tables within the same source, mistakes such as: non-existing customer, account, loan numbers etc, will be detected. Finally, in case of consistency test against other source systems, things like determining the most updated records will be performed.
- De-normalization and renormalizations. De-normalization process will apply to data which will be pulled into dimensional data model in central data warehouse database to be able to match the dimensions. In some cases, data has already been de-normalized, it is then necessary to renormalized it to recovery parts of the records.
- Data Type Conversion. This is a simple action in which old data format used in the source systems will be converted into another to match the format in the data warehouse system. Examples of this action can be: converting 12/25/2001 to 2001-12-25, converting EBCDIC character set of IBM's mainframe system to ASCII etc.
- Data Aggregation. This is a step can be done in both data transformation and data loading process, which will be discussed in detail next subsection. The goal of data aggregation is to provide cost-effective query performance.

By the end of the data transformation, data should be:

- Consistent by all measures
- In required data format, i.e. in required data type, length, style for dates and numbers
- Error free
- In required aggregation

Therefore, the waiting-to-be-loaded data should be in a structure that will speed up any future queries. Many commercial tools can support data transformation.

2.4.3 Data Loading

Data loading is a comparably easy process. It is simply to place the cleaned and transformed data to the place they belong. There are two major concerns here. The load process must support for multiple targets and able to load large volumes of data efficiently.

- Support for multiple targets. According to their usage and role in future queries, raw data, aggregated business data and metadata, which we mentioned earlier, should be placed into different databases and in different syntax. The loading process need to put these issues into consideration.
- Support for large volume of data. Depending on the size of the business, one data load may involve tens of GB, or even TBs of data. Some loading approach used to quicken the process is suggested by T. Barclay. In [6], a parallel loading approach using dataflow parallelism was proposed to speed up the loading process.

2.4.4 Data Refreshing and Data Purging

The processes, data refreshing and data purging, will be discussed in this section. These topics are being neglected topics in many books; however, we feel it is important to discuss these processes and pay attentions to them as to any other processes, since they too fall into data maintenance category.

After the initial loading of data, the updates of the source databases should be propagated to the data warehouse. This propagation process is called data refreshing [7]. Like other processes above, data refreshing has its own issues that one should consider: consistency requirements, refreshing timing, the modes of refreshing and refreshing techniques.

- Consistency Requirement. According to the requirement of the application, level of consistency is different.
- Refreshing timing. It is to decide "when to refresh". Obviously, it may occur right after every single update at the source, or one can choose to refresh periodically, e.g. daily, weekly, monthly etc. Furthermore, the timing can be set by the end users in special events. It is mainly based on the requirement of the application.
- Modes of refreshing. Two modes of refreshing are available, online refreshing and off-line refreshing. They are separated by whether the applications are kept running while refreshing process is performed. As one can guess, online refreshing is the one which applications run at the same time refreshing is processing. The danger of this mode is that data inconsistency may occur since user might perform query on any data

stored in an aggregated table while its counterpart in raw data has just been updated. Off-line refreshing is considered much safer in keeping data consistent; however, it will stop all applications until the end of the process. Off-line refreshing is not suitable in business which requires applications running 24-7.

Data Purging is a process in which old data in data warehouse is removed, cleaned and archived. The purpose of data purging is to reduce the amount of stored data in the data warehouse. The definition of the *old* data depends on the regulations of the organization and the requirements of the applications. In most cases, these *old* data refers to data that is no longer valid, or rarely used. The purging policies include [7]:

- Cleaning. Old data is 100% removed from the DW, including atomic and aggregated data, but the metadata about the old data remains in the DW. If there is need to refer to the purged data, the system will be able to locate where the operational data resides. Data reloading can be either manual or automatic.
- Selective Purging. Old data is selectively purged as a function of space, access frequency and performance requirements.
- Archiving. This policy stores old data on archiving storage devices, e.g., tapes, or other disks, before purging them. Basically, only the most detailed data is archived, since aggregates can be derived from them. This policy is used when the operational DB systems or the legacy systems are not able to provide historical data any more and the historical data will be accessed sporadically.
- Selective Purging and Archiving. This policy is the combination of Purging and Archiving. In the cases that the operational systems do not provide historical data and

the access frequency to the old data is relatively high, we could archive the detailed data onto a tape before purging it and selectively leave some aggregates in the DW.

The choice of applying one or more of these policies depends on the cost benefit analysis of maintaining the data warehouse.

Chapter 3

Data models and OLAPs

3.1 Data Modeling

Two types of data model are widely used in constructing a central data warehouse. They are relational data model (RDB) and multidimensional data model (MDB). There has been argument for a long time which data model is the key to a successful data warehouse. Multidimensional data model supporters think relational data model is hard to understand and with poor response performance [3, 4, 5, 12, 17, 27, 33, 37, 38, 39]. Relational data model supporters are focusing on the sparsity problem of MDB [2, 11, 28]. It is not this thesis's focus to discuss which type of data model is superior comparing to the other in general. Instead, we focus on showing that the choice of data model that is used for central DB in a specific project should really depend on the main characteristics of the central data warehouse.

For a data warehouse that needs to store large amount of data for business analysis, relational data model is a better choice. Relational data model uses the concept of relation as its data structuring concept [26]. The main advantage of relational data model is its capacity to store large amount, detailed, atomic data. On the other hand, if the amount of data for analysis purpose is relatively smaller and emphasis is placed on the performance of DB, a multidimensional data warehouse would be more suitable. This is because MDB is focusing more on the performance and application of the data warehouse. It is more "cognitive", easy to understand by end users and implement business rules for DB designers. Because of their different characteristics, their applicability is rather different. It is therefore improper to

generally compare the two data models. The discussion should be limited to individual cases. In a specific project in which the conditions and objectives are clearly defined, it is then possible to determine which model is better suited for what purpose based on the advantages and disadvantages of the two data models.

Each of the above data model has its unique advantages. The application of the two data models in one project is not mutually exclusive. On the contrary, relational and multidimensional data models are quite complimentary to each other. In reality, it is hard to find a project that doesn't require large amount of data for analysis. At the same time, fast query response performance is equally desired in building a data warehouse. Therefore, the hybrid architecture -- having a relational DB for central storage in the DW, and then transfer data into multidimensional in presentation layer, is winning the popularity contest over single-model designs for data warehousing projects. We will show a design that incorporates this hybridization in the later sections of this chapter.

3.2 OLAP Technology

OLAP (Online Analytical Processing) is gaining popularity rapidly in today's business world. According to the OLAP council [10], "OLAP enables end-users to perform ad hoc analysis of data in multiple dimensions, thereby giving them the insight and understanding they need for better decision making. Its added intelligence about the structure and organization of the data, as compared with flat, detailed relational tables, makes an OLAP server more responsive to end user requests while also eliminating SQL-style queries." Codd, et al. defines twelve rules for OLAP products [14]. Other references on OLAP can also be found at Data Warehousing Information Center [15].

There are two fundamental methods of realizing OLAP; both are based on multidimensional view of the data. They are Multidimensional Online Analytical Processing (MOLAP) and Relational Online Analytical Processing (ROLAP), each with its unique advantages and disadvantages [40]. There is no generally accepted conclusion that one is superior compared to the other. Application of either method in any given project largely depends on the characteristics of business requirements and goals.

In recent years, Hybrid OLAP (HOLAP) is becoming an interesting alternative to combine the strengths of MOLAP and ROLAP, (e.g. Microsoft SQL Server OLAP services and Pilot software's Pilot Decision Support Suite). One method of achieving HOLAP is to incorporate two databases. A relational database stores most of the data and a separate multidimensional dataset stores the densest data, which is typically a small proportion of the data. Unfortunately, the method of hybridization varies from application to application. There is no widely accepted conceptual architecture of HOLAP, making the subject more difficult to discuss in general terms. It is therefore excluded from the discussion below.

3.2.1 Multidimensional OLAP

Multidimensional OLAP (MOLAP) utilizes a proprietary multidimensional database (MDB) to provide OLAP analyses. "The main premise of this architecture is that data must be stored multidimensionally to be viewed multidimensionally." [11] Currently available MOLAP tools on the market are: Hyperion Essbase, Oracle Express, Cognos PowerPlay (Server) etc. A basic schematic of MOLAP is shown in Figure 3.1.

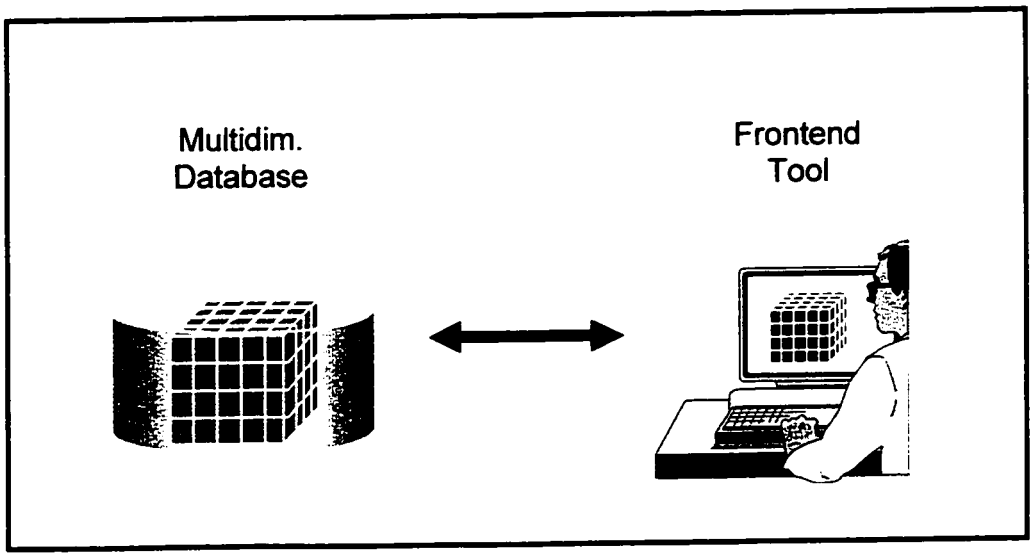

Figure 3.1: Schematic of MOLAP

The basic characteristic of MOLAP is that in MOLAP data is kept in array storage, which makes it inherently multidimensional [17, 20, 21, 40]. The design of the MDB shall reflect business rules. The database itself and the data structure are "cognitive" to end users. That is to say any end-user, who is familiar with the business operation but not trained in DB, can easily understand the data structure and perform his desired analysis or query directly. Operations such as roll up, drill down and rotation, slice and dice are automatically implemented. These operations can be performed with great speed according to predetermined aggregation rules.

Once atomic data has been loaded into this multidimensional database, it will automatically perform a series of calculations to aggregate along the dimensions in different level, and the results will fill the structured array [18]. This can be shown by following example:

Table 3.1: Aggregation along dimension in MOLAP

Day	Month	Quarter	Year

Day 1	January	Quarter 1	2002
Day 2			
Day 3			
....			
Day 31			
Day 1	February		
Day 2			
Day 3			
...			
Day 28			
Day 1	March		
Day 2			
Day 3			
...			
Day 31			
...	...	...	...

Table 3.1 is an example of time dimension, which appears in almost every data warehouse since virtually every data warehouse is a time series. The Time dimension in this case has four fields. They are: day, month, quarter and year. Once the data loading of corresponding days, the atomic data, is completed, MOLAP will automatically aggregate the data in a way such that the results of Month are the sum of days' under this month; the result of Quarter 1 is the sum of the numbers of all three months; and Year 2002 will be the sum of all four quarters.

After the loads and aggregations of data, indices are created and hashing algorithms are used to improve query access times. Once this compilation process has been completed, the MDB is ready for use. Users request OLAP reports through the interface, and the application logic layer of the MDB retrieves the stored data.

Strict MOLAP architecture, or a model textbook MOLAP architecture, would use MDB as both the database and the application logic layer. However, this is not always what people do in real life. A different architecture using MOLAP server with a case study will be presented in

the following sections. MDB here is responsible for both database layer functions such as data storage, access, retrieval processes, and for application logic layer functions such as execution of all OLAP requests.

Compared to other types of OLAP architecture, MOLAP is relatively easy to design, apply and maintain. This is because the basic design of the architecture and the data model are based entirely on business rules and operations. In other words, the architecture offers a good conceptual fit with the way end-users visualized the business data. As long as data is successfully sorted, cleaned and loaded into the MDDB with the business-oriented data model, data will be automatically aggregated and ready to be used [19]. With MOLAP, every aggregated data is stored in the array. This way, end users have direct access to array data structure, and are able to view their desired data in an instance by rolling-up and drilling-down along the dimensions, and/or slicing and dicing the cubes to whichever ways they feel more comfortable with.

Attractive as its advantages are, MOLAP has a few noticeable disadvantages. First of all, the amount of data a MOLAP can handle is rather limited. Many studies have shown that the amount of data handled by a MOLAP server should not exceed 200GB. Text below in the box is taken from [11], it shows how big in size a MDB could be when the real-world input was only 57MB. The fact is, the size of a MDB increases in an exponential ways.

Expansion in a Multidimensional Database
In one benchmark published by a multidimensional database vendor, an eight-dimensional data model with a 57-megabyte sample of real-world input information expanded to 43 gigabytes in the final multidimensional database.
While 57 megabytes is a useful test sample, a larger volume of input information fully illustrates the limitations of the multidimensional approach. Imagine the database size with 5 gigabytes of input data.[11]

Secondly, performance will be best when the number of dimensions of a hypercube is less than 8. Although theoretically there should be no limit on the number of dimensions forming a multidimensional cube, in practice data cubes with dimensions higher than 8 will not only make the results difficult to interpret, but also decrease the overall system performance. Finally, since the structure of the multidimensional cubes is fixed during the design phase and the number of dimensions of a cube should be no more than 8, the viewing point is fixed for the end users. If any end user wishes to see different view points other than those predetermined in the design process, a complicated operation has to be performed. For example, if a user wants to see a dimension from cube 1 and two dimensions from cube 2 and one dimension from cube 3, this will take longer time to process and sometimes even impossible for an end-user to complete this task. This action is called "drill-cross". It is slow, complicated and requires high level of knowledge and experience.

The benefits and short comes of MOLAP can be summarized into following points:

Advantages:

1. Cognitive Advantages for the User
2. Ease of Maintenance

3. Ease of Data Presentation and Navigation

4. Quick Reaction

Disadvantages:

1. Limited data handling

2. Limited dimension handling

3. Fixed viewing point due to pre-set cube dimensions

3.2.2 Relational OLAP

Relational OLAP (ROLAP) accesses data stored in a data warehouse to provide OLAP analyses. Unlike Multidimensional OLAP (MOLAP), "The premise of ROLAP is that OLAP capabilities are best provided directly against the relational database, i.e., the data warehouse" [11]. Typically there is an OLAP middleware to support user interaction with the database. In ROLAP, there's no direct user access to stored data. This design feature provides better security to the database. Currently available ROLAP tools on the market are: MicroStrategy, SAP BW etc. Schematic of a typical ROLAP system is shown in Figure 3.2.

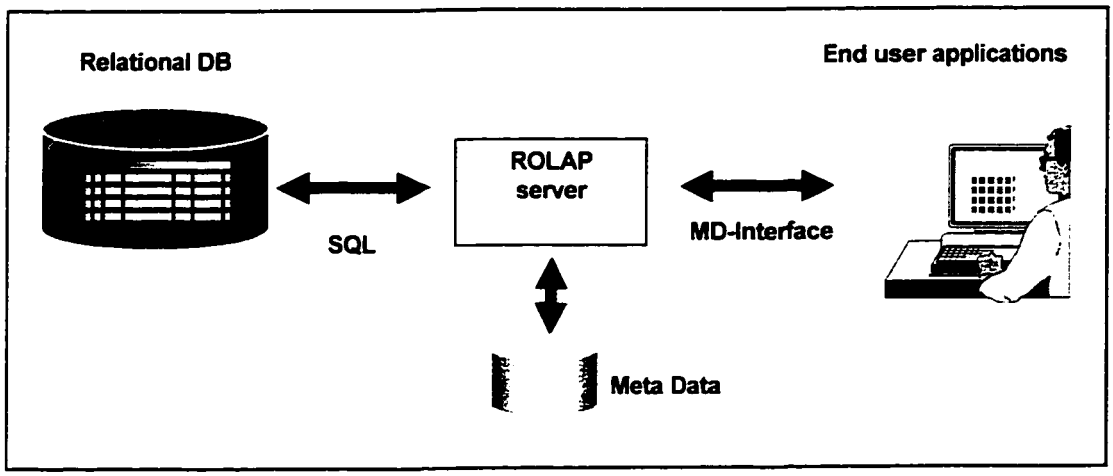

Figure 3.2 Schematic of ROLAP

ROLAP is a three-tier, client/server architecture; this architecture includes a central database, a ROLAP server and the end user applications layer, or the presentation layer. The database layer utilizes relational databases for data storage, access, and retrieval processes. The ROLAP server, as the logic application layer, executes the multidimensional reports from multiple end users, and also integrates with a variety of presentation layer applications, through which users can perform OLAP analyses.

In general, once cleaned and transformed operational data is successfully loaded into the central DB, preset database routines are called to aggregate the data. In some of the hybrid architecture, however, there can also be a secondary transformation to be performed on the data so that it will be ready for viewing in a multidimensional way at later stages. This will be discussed in more details in the section following the case study. For now, we will only focus on the first scenario.

After the data is loaded into the database, Indices are then created to optimize query access times. Unlike MOLAP server which simply goes through the MDB searching for the answer

to the requests made by the end users, ROLAP server is able to dynamically transform end users requests for multidimensional analysis into SQL executables for those queries that are not pre-calculated. The SQL routines are then submitted to the relational database for processing, and the results will be returned to the end users in multidimensional view. The end users of the ROLAP system will not see any difference between the multidimensional results produced here and any other OLAP systems. However, there does not exist any physical cubes like those of a MOLAP system. In ROLAP there are only virtual cubes. By using these "virtual cubes", ROLAP becomes a fully dynamic architecture. It is flexible in cross examining different dimensions even when they are seldom related, by dynamically generating results from atomic information taken directly from the RDB; moreover, it is also fast to produce the analytical results by utilizing pre-calculated data stored in aggregation tables.

Other than being flexible in answering queries and fast response time when results are pre-calculated, the most attractive advantage of ROLAP is the data handling and dimensional handling ability. Because data is stored in relational way, there is only minimal redundancy. Unlike MDB in the previous section, the amount of atomic data in relational DB is the amount of input. That is to say there is little or no increase in the amount of data in RDB. Therefore, given the same available size of storage, RDB system can store much more data than MDB. In addition, RDB is also able to store large amount of pre-calculated data in forms of aggregation tables for future fast access. Without having physical multidimensional cubes, ROLAP is able to handle more dimensions than MOLAP does, and the ways of arranging these dimensions are not fixed.

Finally, with ROLAP, end users are not able to write into the central database as in MOLAP. This gives two advantages over MOLAP: security and backups. Since everyone is only working with a copy of data for analysis in ROLAP, no one is able to change anything in the central database. In case anything happens to the copies of the data, or front-end application crashes, the data in central database of the warehouse is well-protected, and DB routines can lately regenerate pre-set aggregations.

Like almost everything, ROLAP is not perfect. It too has its disadvantages. First of all, in order to design a usable ROLAP system, the designers will spend more time in requirements analysis. The design in MOLAP is quite straight forward by simply following the business rules to build the structure of the warehouse as in how people understand the business. ROLAP design will first have to build a central relational DB and then all possible aggregation routines. The results of these aggregation routines should be able to cover most of the everyday queries and reports. Although these queries and reports may be covered in the requirement documents from the clients, as any experienced system designers know, a requirement document from the clients is rarely clear about what exactly the client wants and how he wants it. In addition, the system designer himself is likely not an expert in the business area of the client. As a result, it is very hard to foresee all possible questions in the future even with the detailed requirements from the client. All factors considered, it takes a lot of efforts in designing a successful ROLAP system.

Complicated design is only the first step towards a successful ROLAP project. Once successfully designed and implemented, the ROLAP system is hard to maintain. This is mainly because of its dynamic viewing ability. As mentioned previously in this section, one of the

advantages of the ROLAP system is its flexible view point. The end-users are not able to make direct change to the database, but they can view the data various by accessing the pre-calculated results and in some cases having copies of data made for answering queries. Other than atomic data, RDB is also saving some pre-calculated aggregation results which should cover 95% of all the queries and reports. But, if the direction of business changes or the structure of the organization changes even slightly, it may affect the ways end users want to see the data. There are also times during which the request to the remaining 5% queries and reports that usually are infrequently visited increases suddenly. To meet the changes in business nature or organizational changes, new aggregation routines need to be added and old ones removed constantly. Therefore, the system administrator of the ROLAP system has more maintenance tasks than that in MOLAP system.

Finally, the response time of a ROLAP system can be sometimes slow. MOLAP systems do aggregations automatically in all levels and directions whereas ROLAP systems do aggregations selectively according to the requirements of the clients. For a query which is not in the list of pre-calculated aggregation results, ROLAP engine will have to start from the scratch to be able to produce the result. The process steps, that translate the query into SQL, access the relational DB for data and execute joining/grouping algorithms, take time. The response times for these types of queries and reports in ROLAP are significantly longer when compared to similar operations in MOLAP. This is unfortunately the sacrifice we have to make in exchange for the flexibility that comes with ROLAP.

The benefits and short comes of ROLAP can be summarized by following points:

Advantages:

1. Able to handle large amount of data

2. Able to handle many dimensions
3. Flexible view point
4. Secured database, and backups

Disadvantages:

1. Hard to design
2. Hard to maintain
3. Slow response time when results are not pre-calculated

With ROLAP, clients are able to pre-set aggregation levels and regions. During queries, the ROLAP server is able to direct the search to the level that is closest to what the clients are asking.

For example, if the Time dimension has 5 fields day, week, quarter, month and year, but only quarterly, monthly and yearly aggregation data are accessed 95% or more of the total time by the end users. These three levels' aggregation would be set during the design phase. If at some occasion when end user wants to perform a query about "total sales made in the first week of March, 1999", a complicated process has to be performed before correct answer can be given. To search for the answer, ROLAP server will first browse through the smallest grain of available aggregation data, which is quarterly aggregation, and then go up one level and up again until reaching the last aggregation level on this dimension. When no answer to the query is found, the ROLAP server will direct front-end applications such as Brio [16] to automatically translate the query into SQL that will calculate the answer based on the data stored in the central RDB. This process will be very time-consuming. The only source of

comfort is that queries like this will only be less than 5% of the total. It is neither necessary nor wise for ROLAP to keep a lot of storage space for queries that only happen 5% of the time.

On the other hand, if end user asks for "total sales made in March, 1999", ROLAP server would then lead the search directly to the monthly aggregation data. This search is no slower than that in MOLAP system.

Chapter 4

Case Study --- B Mobile Communication Co. Ltd

4.1 Background

B Mobile Communication Co. Ltd offers mobile communication and internet services to Beijing and near-by areas since July 26th, 2000. There are total of 4.5 million customer registered with B Mobile Communication as of today. This number will reach 6.0 million by 2003. Approximately 250GB of data per day is generated by customers. Within this number, transaction data amounts to approximately 200GB per day. Total amount of transaction data of the biggest fact table alone is around 2TB per year. It is very necessary and beneficial to analyze these transaction data to study the business patterns and discover hidden unexplored market segments.

Let us consider the following example. Client A and B are both late on their payments. Standard corporate policy for late payment is to stop the service until the balance is paid in full. Upon closer examination, we may find that Client A has a perfect credit history, always paid his balance in full on time, and has high level of monthly usage. He is the type of customer that the service provider would love to keep. It is risky to alienate this client by suspending his service for a single occurrence of late payment, which may even drive him to the competitors. It would be wise to have a customer service rep to inform him of this late payment issue and not suspend his service. On the other hand, client B rarely paid his balance on time and is considered a risky customer. His service should then be suspended until his balance is cleared.

On a higher level, business can use the cumulated data to add new service, explore new market segments and increase its market shares. For example, analysis shows that business users tend to use enhanced voicemail, in combination with text messaging. It is then possible to offer a combo of these two services to appeal to these users. Since the fee for the combo is less than the total of the fees for both services and only slightly higher than each, it will definitely attract customers who initially only want to have either one of the services. In addition, promotions and advertisements can be more focused to potential customer groups. It is easy to understand that ability for international roaming is necessary to business travelers, while newer, smaller, more functional handsets are more appealing to younger generations. These cases are only simple things a business can do with Business Intelligence solutions.

In order to stay ahead of competitors and provide better and more accurate services to its clients, B Mobile Communication Co. launched this BI project in which a data warehouse will be built for data analysis purpose. The users of this project will from three departments: Customer Service, Marketing and Sales.

For business reasons unrelated to this project, the hardware selection is already decided before any suggestions of the architecture were actually made. They are: one IBM Enterprise Storage Server, also known as Shark, of 5TB storage space, two IBM RS/6000 S85s. The total storage space is also pre-decided without doing any research and actual calculation. Within this project, the tasks of requirement analysis and design of the overall DW model are given to Global Business Intelligence Consulting.

Global Business Intelligence Consulting Co. (GBICC) is a consulting and system integration firm, specializing in Business Intelligence (data warehouse, data mart, and data mining),

analytical Customer Relationship Management (CRM), risk management, etc. GBICC also develops special applications of the above mentioned fields of interest. So far it has formed alliance with IBM, HP, AsiaINFO and Digital China. Detailed information is available at www.gbicc.net.

BJ Mobile's internal data source for this project is its Business Operating Support System (BOSS). BOSS developed by Prosten Technology (www.prosten.com), is a powerful, enterprise-class information communications platform. It is integrated with the company's Management of Information Systems (MIS) and Enterprise Resource Planning (ERP) systems to provide instant access to vital business information and key enterprise data that will help expedite the business decision-making process.

The BOSS system consists of four major functional parts as illustrated in Figure 4.1.

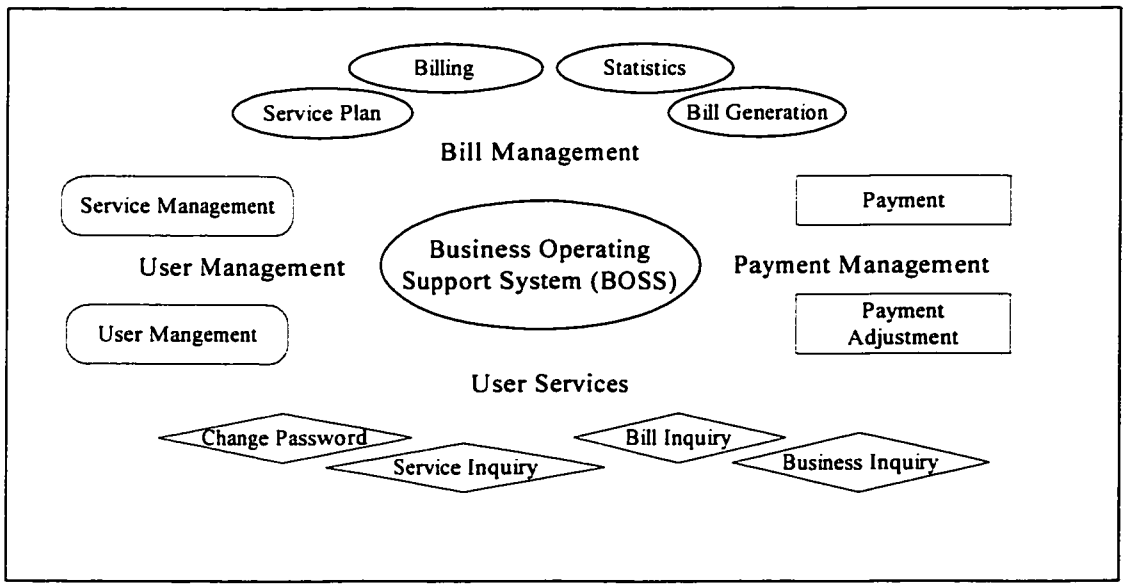

Figure 4.1: Functional Composition of BOSS [13]

1. Billing management

Billing management module is the core of BOSS. It is the back-end engine to support business management and payment management. It includes developing service plan, billing strategy and automatic bill generation according to billing period. The input of billing management is the marketing plan of the telecom operator that specifies types and contents of value-added services offered, relevant billing strategy, preferential treatment as well as penalty.

2. User management

The customer model orientates the entire user management module. A reasonable design of customer model dictates the service providers' business management process directly. User management includes customer information management and service management. Customer information management, the first step of service management, has the following functions: opening account for new customer, authentication, authorization, account cancellation, information inquiry and alteration. Service management generates order based on customer's service provision, then tracks and manages the order.

3. Self-service

Self-service is an important feature in BOSS. Users are allowed to check their own service information at the service provider's website. User service information includes:

- Inquiry of service instruction
- Bill inquiry
- Service provisioning inquiry
- Change password

4. Payment management

Payment management module manages payment by customer, whose bills are generated by billing management module.

At B Mobile, BOSS consists of two sub-systems, each recording different business/operational data, one for Billing_Payment_Management and the other for Client_Management. These two sub-systems are currently running on two separate HP V2600s. Client_Management data are stored on Oracle DB, and Billing_Payment_Management data are stored as binary files. Target data will be from these two sub-systems.

The definition of a successful data warehouse is highly individual. Before different designs for this project are presented and their merits discussed, criteria to evaluate them must first be established. The client of this project values maximum query performance, wide range of queries the system can provide to end users, storage capacity, flexibility, and potential for expansion and upgrade on storage capacity and processing speed, etc. These properties, however, are very difficult to measure quantitatively in any scaled-down model. A typical example is the performance of multidimensional database. As mentioned in previous chapter, MDB performs better when the data dimension is less than 8. This fact can only be verified when the performance deteriorates with data cubes of higher dimensions. It is difficult to conceive a scheme to simulate the deterioration of performance with increasing dimensionality of the data cube in a scaled down model with small amount of data. In addition, there are no generally accepted methods or parameters that can translate the performance in a scaled down model to credible projected performance of a fully implemented data warehouse. For these reasons, it is impossible to perform any simulation on the designs to compare their performances. At this stage, we can only rely on the specifications of the software and hardware, some limited theoretical analysis and expert opinions to make our decisions on the design.

4.2 First Design

The first design of the warehouse architecture was proposed by company A. The software and hardware for this project had already been chosen at this stage. This design was made to fit the hardware and software already chosen by the client. Fig 4.2 shows the components of the first warehouse design.

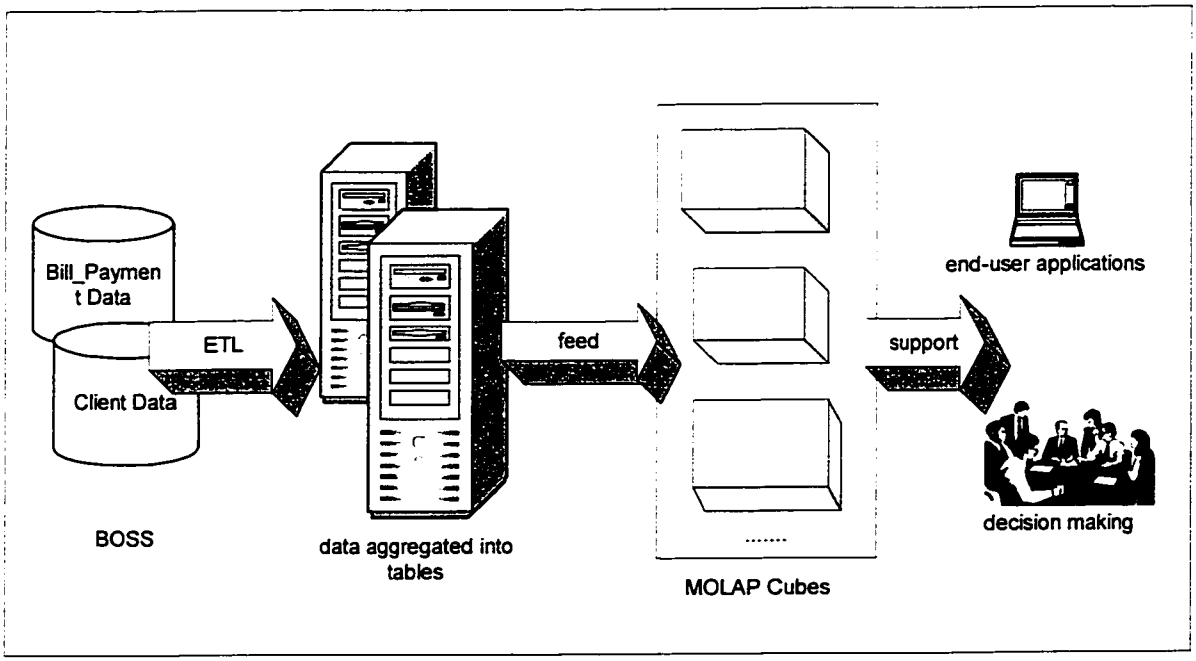

Figure 4.2: Designed by Company A

The source data of the warehouse is from the BOSS system. They will pass through the ETL processes (data extraction, data transformation and data loading) into the aggregation table, which is the center of this design. Then, several multidimensional cubes supporting MOLAP server will access this aggregate table. In another word, the aggregation table feeds the cubes, which will serve the end user applications such as query and reports. Decision makers will then be able to make business decisions based on the analysis results.

Figure 4.3 shows the schematics of the first design.

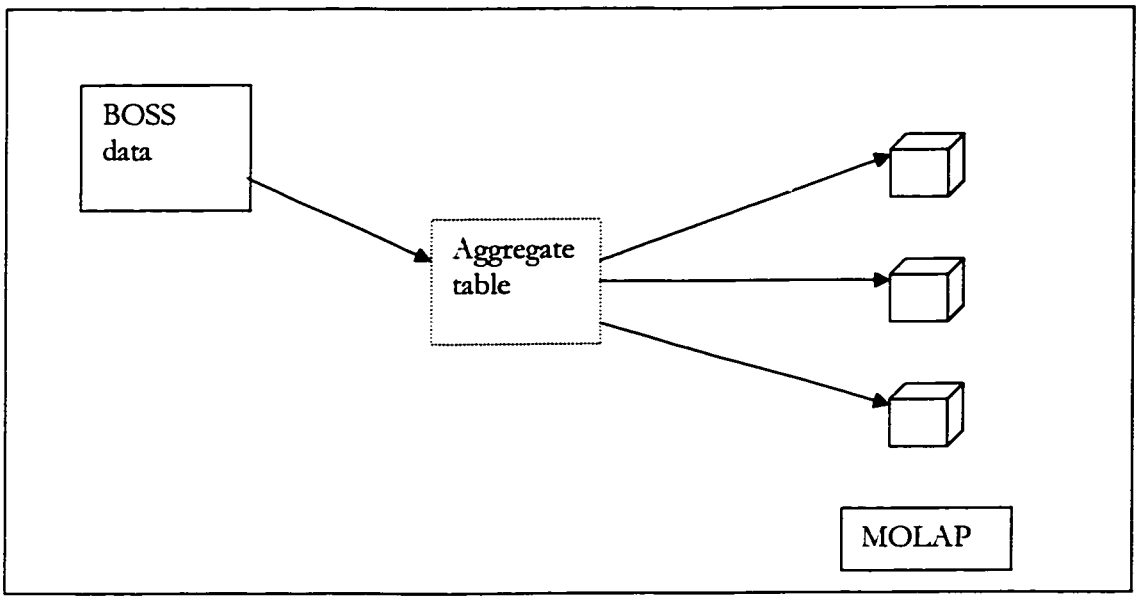

Figure 4.3: Schematic Design by for Design 1

The data aggregation process is shown in the following example:

*following example is used just to show some possible changes made to data records. It is not taken directly from B Mobile case.

Table 4.1: Client Information in BOSS

Client No.	123456	234567	...
Client Name	John Woo	Jacky Chen	...
Client DOB	August 13, 1943	October 12, 1965	...
Client Sex	M	M	...
Identification type_ID	1	1	...
Identification No.	35927502850482	15354357656774	
Client home address	123 some street	456 Lemon street	....
District	Chao Yang	Chao Yang	...
City	Beijing	Beijing	...
Postal Code	100027	100027	...
Registration Date	July 23, 2001	May 2, 2000	...
Client Status_ID	1	1	...
...	...	...	...

In BOSS system, information of clients is recorded in details. Table 4.1 is an example of partial client table with detailed client information recorded. Before being loaded into the aggregation table, the data in this table will have to pass ETL process. In the ETL process, useful data records will be extracted from the source database, different data type will be transformed into unified data type as in the aggregation table and undesired information will be removed or cleansed. Finally, well prepared data will be loaded into the aggregation table of the destination database.

Attributes of the client table are changed into the following in the aggregation table:

Table 4.2: Client Information in the Aggregation Table

AGE RANGE	55—60	35—40
Client Sex	Male	Male
District	Chao Yang	Chao Yang
City	Beijing	Beijing
Postal Code	100027	100027
Registration length (in month)	13	26
Client Status	Regular	Regular
...	...	...
Total	2894	167

In the aggregate table, there is no single individual. Everyone is treated as a unit, which belongs to certain range. In this case, Mr. John Woo will fall into a group which has following characteristics: age range is between 55 to 60, male, living in Chao Yang district, postal code is 100027, has been registered for 13 months, is a regular client... etc. Unlike Mr. John Woo, Mr. Jacky Chen falls into a group which is between 35 to 40 years old, male, living in Chao Yang district, with postal code 100027, being with us for 26 months...etc. Individual

characters such as client name, street number and street name, citizen card number etc are eliminated. Client DOB and exact registration date are calculated and fall into different ranges.

At the end of one aggregate table, there will be a "total" count. This is counting the total number of clients within this measuring group. Table 4.2 shows that there are total of 2894 regular male clients, between age 55 and 60 and total of 167 regular male clients, between 33 and 40, are living in Chao Yang district.

In this design, the main data storage is the aggregate table which will support several multidimensional data cubes. The aggregate table and cubes will be updated periodically. However, this architecture design was not implemented in the actual project because of its limitations as discussed below.

Limitations of the First Design

1. This design is to use IBM's MOLAP customer-end server. Problem with this MOLAP design is that the actual data amount is too big (>1.87TB per year). The MOLAP server planned here can only handle 200 GB properly.

2. Without having a relational database, loading and aggregation from BOSS to the aggregated table will have to be performed every time new requirement arises. For instance, when one more department needs to have its own cube for data analysis, data load from BOSS to aggregated table has to be performed again just for this new cube. For the same reason, re-loading has to be done for any updates made in BOSS system.

3. With this design, users are not able to perform detailed data analysis. There are no detailed data saved in the aggregated table. From the previous example of the data aggregation,

one can see that there is no atomic or detailed information saved in the data warehouse. However, not all queries and/or reports are asking summarized questions such as "How many male clients between age of 55 to 60 are living in Chao Yang district?" and "On what average length does these clients use our basic services?" etc. There may be occasions that some end user will ask questions such as: "What are the names of the customers who own more than one cell phone and live on Lemon street?" or "List the addresses of customers who are delinquent and living in Chao Yang District." With this design, the answers for these questions will be very hard to find since no detailed information is saved either on the aggregation table or in the data cubes.

A new design is therefore required to resolve the above limitations. GBIC is chosen to propose a revised design using the same hardware and software.

4.3 Revised Design

The revised design starts from the same BOSS as the design described above. In the previous design, the BOSS data is processed and the results are stored in the forms of aggregate tables. Multiple data cubes are then created from these tables to service different business groups with specific business requirements. In the revised design, ER Model and Star Schema are added to the overall system, as shown in figure 4.4. Data from BOSS is loaded into RDB after going through ETL processes. Data in RDB is further filtered through the second ETL process, then the star schema, and results are kept in the form of aggregate tables. A separate server functions as storage for these aggregate tables. From these aggregate tables, multiple MOLAP cubes are created to respond to different business analysis. These data cubes are derivatives of the stored aggregate tables based on different requirements or business rules. Note here that

the MOLAP system here is different from the MOLAP system mentioned in the previous chapter, in which MOLAP system has a single multidimensional cube serving as data storage and support for data access etc. The revised MOLAP system has approximately 20 data cubes.

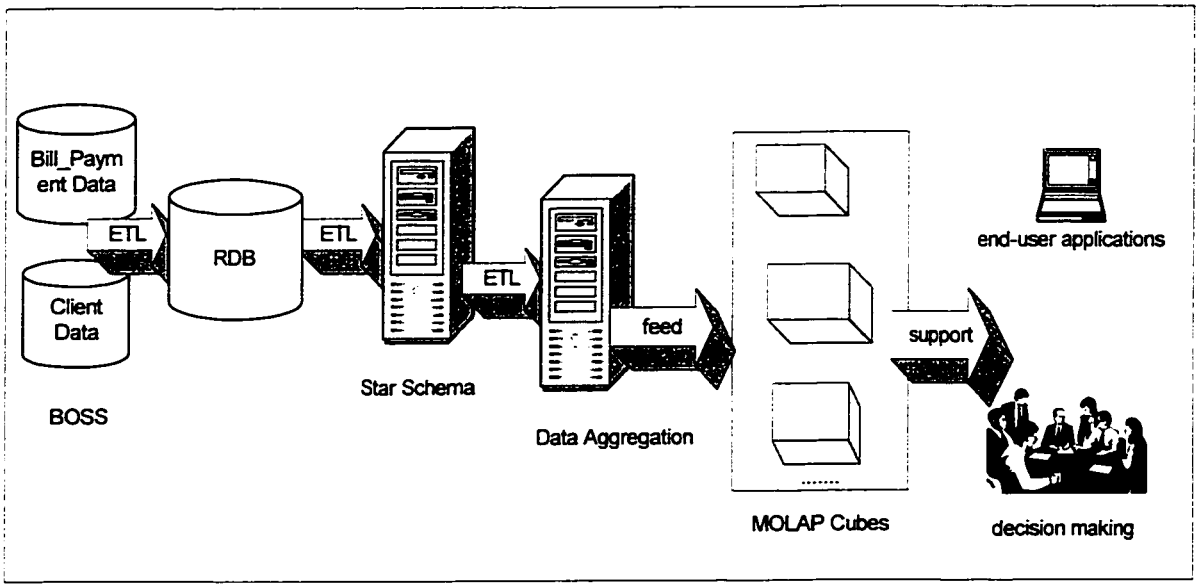

Figure 4.4: Component view of the revised design by GBIC

This architecture model adopts the hybrid design, combining both relational modeling and dimensional modeling. Detailed data coming directly from BOSS are saved as in relational DB first. From the discussion in previous sections, relational DB is good at storing large amount of data without much of redundancy, which is very suitable in this case because data is cumulating at 200GB per day. The second reason for keeping detailed data in a relational DB within data warehouse is because OLAP is not able to satisfy each and every requirement. Even exhaustive study or requirement analysis can not possibly predict all user requests at all times. In addition, the mercurial nature of modern-day business also makes it impossible to anticipate future business requirements. For those analyses involving detailed data which OLAP is unable to perform, end users can use applications such as Brio to realize. (Brio is a

widely recognized software tool by industry analysts. This software package offers three essential functions in business intelligence: enterprise reporting, ad hoc reporting and analysis and global information delivery. More information about Brio software can be found at [16].)

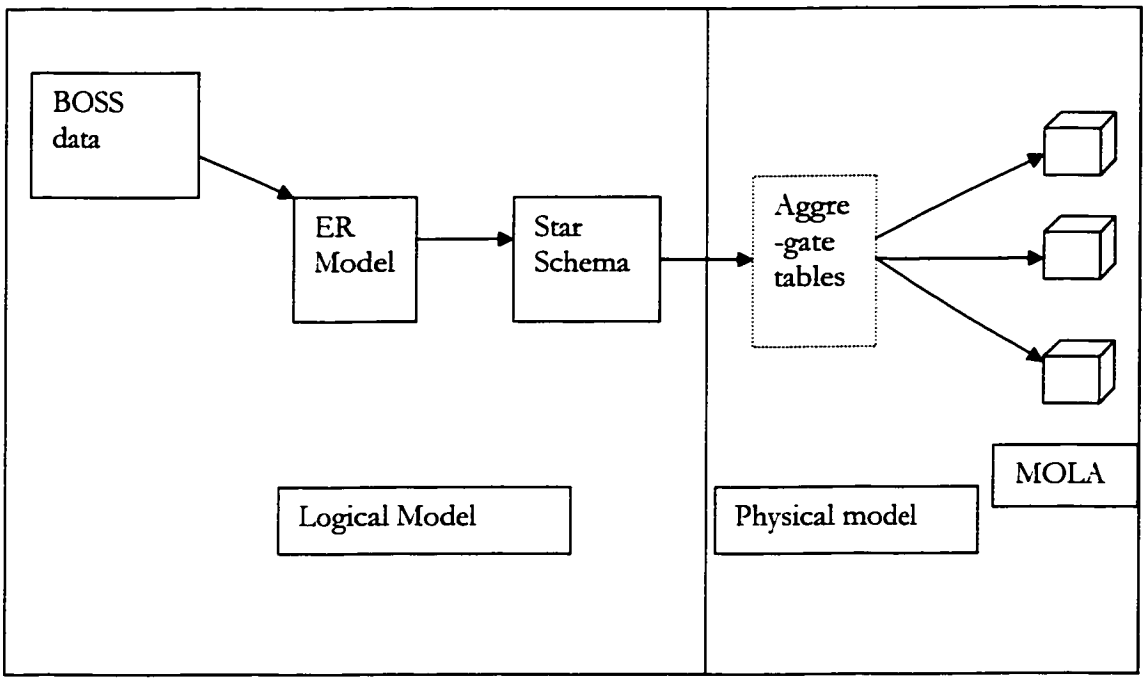

Figure 4.5: Schematic of GBIC Design

After requirement analysis, several star schemas were constructed from the relational DB. The relationship between business area and star schema is in most cases one-to-many, i.e. one business area can have more than one corresponding star schema. Example of such one-to-many relationship from current project is the "Client--Service" star schema. This schema originally covered all services, such as mobile services, messaging services GPRS services and so on, provided by B Mobile Communication Co. The schema included so much information that it was able to support client analysis, service analysis, mobile station analysis, mobile transaction analysis etc. This schema also contains the biggest fact table, "Client – Mobile Services". Because this schema is way too big for any system to store and perform any analysis

function, it has to be separated into smaller schemas, "Client – Basic Mobile Service", "Client - GPRS" and "Client – Short Message". The "Client – Basic Mobile service" star schema alone is approximately 1.87TB per year. Even with separation of a big star schema into smaller schemas, it is still too much for MOLAP server to handle. This is because MOLAP server performs best when the cube size is smaller than 200GB, as mentioned above.

To reduce MOLAP server's work loads, the big schemas such as "Client – Mobile Services" will be further divided into even smaller cubes. In this design, we add aggregate tables in between star schemas and cubes to do just that. Theoretically, these additional aggregate tables are not necessary. The reason to have these extra tables is to speedup the aggregation and loading processes between star schemas and final cubes. Each cube contains data aggregated under different rules and is used to answer one specific category of questions regarding certain aspect of business operations.

The granularity of star schemas is set according to the requirements. Since questions are related to clients in most analysis cases, the smallest granularity, i.e. the atomic level, is set to individual client. Keeping data in such small grain may also be very helpful in making future system changes. The data warehouse system is a "forward" system, which is to say that the data flow is going forward. The data will not be able to recover once it has been modified. Therefore when any future change needs the atomic level data, it does not have to execute the first ETL process to generate these atomic data from operational data in BOSS. Instead, the atomic level data is readily available in RDB. This design provides great flexibility and expandability for any future modification or addition to the system. In the previous design all stored data has been aggregated to a certain level, there is no atomic level data available. If

there is new demand for data to be aggregated different than those stored, it will have to recreate atomic level data from BOSS through new ETL processes.

The aggregation table is the same as it is in the previous design. The main difference between these two designs is in the data flow. In the first design, the data is directly loaded from BOSS and goes through aggregation process. The results are stored in the multidimensional cubes. In the revised design, the data source is still the BOSS. The data is loaded into RDB through ETL before any aggregation is attempted. From RDB, the data is then filtered by the Star Schema. The results from Star Schema will undergo another process of ETL and the end results become the aggregate tables.

Data Formation Process

In ER model, many attributes are recorded using Id numbers. This differs from the recorded data in Star Schema.

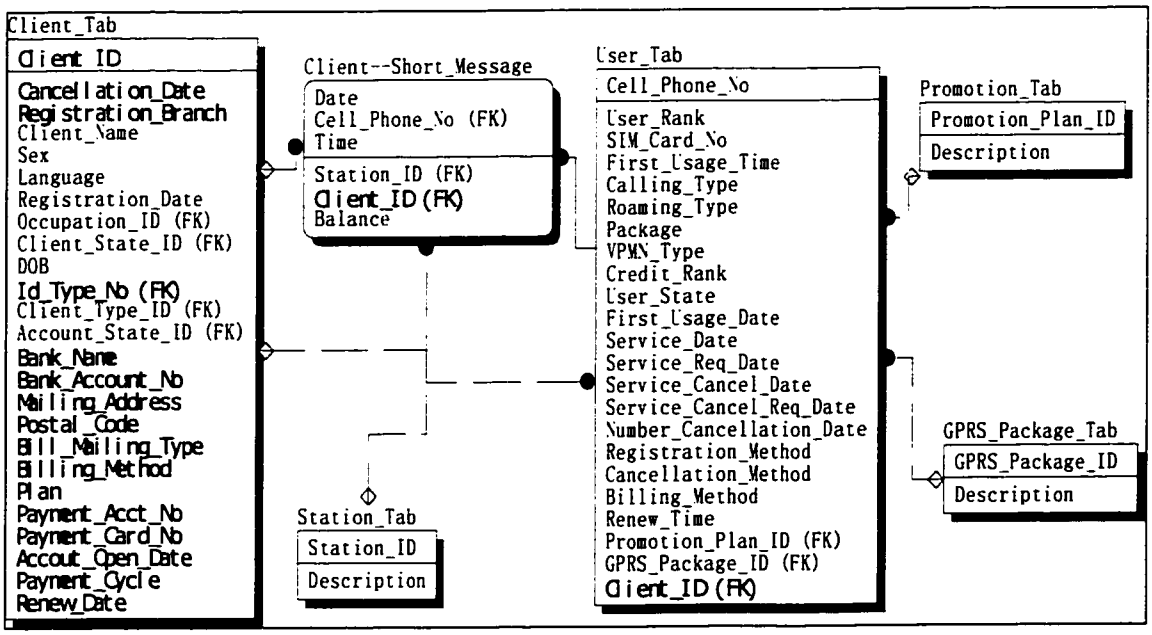

Figure 4.6: Client—Short Message ER Model [From B Mobile Project]

This figure is the actual partial ER model of "Client – Short Message" taken from B Mobile Project. The corresponding star schema of the Client – Short Message is presented bellow.

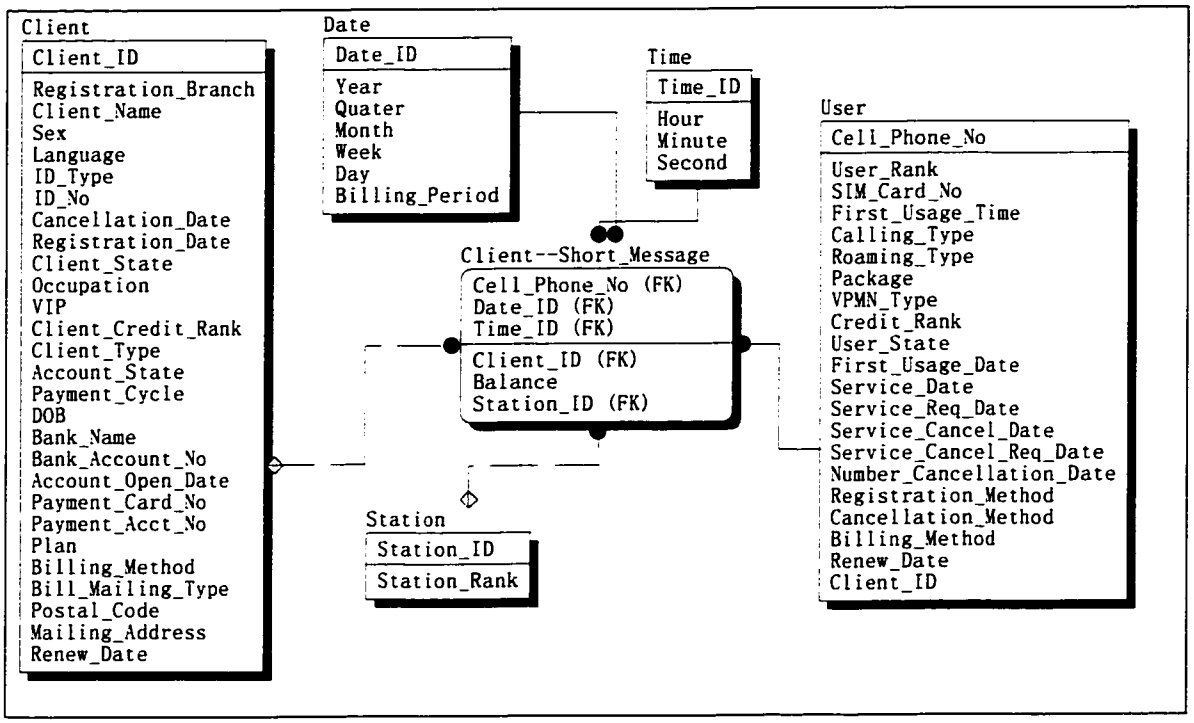

Figure 4.7 Star Schema of "Client—Short Message" [From B Mobile Project]

Contrary to the conventional wisdom, the differences between the ER model and star schema of "Client – Short Message" in this design are actually quite limited. The two models are intentionally kept similar by design. This is to reduce the amount of transformation from RDB to Star Schema. Taking a closer look at these two models, the differences between figure 4.7 and 4.6 are:

- addition of Time dimension in Star Schema
- central fact table, making all access from dimension tables symmetrical
- removing unrelated tables according to the business area

All these differences are making the star schema multidimensional. However, we can see that starting from the ER model it is already very "business-like". It follows that ER model can be designed to follow business rules to some degree. By this design, it is very helpful in transforming the ER data into multidimensional data, whose purpose is to present the business operations.

Although their overall structures are quite similar in design, the data in these two models are quite different. Taking the Client table in the ER model, and Client dimension in the Star Schema as an example, a Client record in ER model would be something like:

Table 4.3 Client Record in ER model

Client_ID	123456
Client_Name	John Woo
Sex	M
Language_ID	1
Registration_Date	July 23th, 2002
Occupation_ID	3
Client_State_ID	1
DOB	August 13th, 1943
Id_Type_No	1
Client_Bank_ID	2
Client_Type_ID	4
Account_State_ID	2
....	....

Most information of a client is saved as ID numbers. E.g. Language_ID 1 = Chinese, 2 = English etc; Occupation ID 1 = Accountant, 2 = Actor, 3 = Director etc.; Client State ID 1 = active, 2 = inactive and so on.

The same client record in a star schema is very different.

Table 4.4 Client Record in Star Schema

Client_ID	123456
Client_Name	John Woo
Sex	M
Language	Chinese
Registration_Date	July 23th, 2002
Occupation	Director
Client_State	Active
DOB	August 13th, 1943
Id_Type_No	Passport
Client_Bank	Bank of China
Client_Type	Valuable
Account_State	Open
....	....

The field descriptions have lost the "ID" number indications. This change makes the data more cognitive. The changes made from table 4.3 to 4.4 are once again the result of ETL processes.

It should be noted that in the first design, aggregation is a one step process. Data goes from BOSS to aggregate tables in one ETL process. In the revised design, the aggregation is a multi-step process. The data will have to go through three ETL processes, BOSS to RDB, RDB to Star Schema, and Star Schema to Aggregate tables.

Tables and cubes

The relation between aggregated tables and the cubes can be many-to-many. In this case, one cube may access one aggregated table, and one aggregated table may produce many cubes. The number of corresponding aggregate tables depends entirely on the property of cubes. For example, if two 3D blocks have dimensions like ABC and BCD, it would be helpful to prepare

a four dimension aggregate table, ABCD as long as the size of the cube is within the bound of storage limit. From block ABCD, we can simply do an aggregation on dimension D to yield ABC, and for BCD, we can do an aggregation on dimension A. In this case, we use much smaller storage space than having to store one aggregate table for each block. On the other hand, if two blocks have very little inter-relation between them, for example ABCD and EOJB, it is better to store them as two separate four dimension aggregate tables rather than force them into one seven dimension aggregate table ABCDEOJ. As two separate aggregate tables, it is much faster to load the data and saves a lot of storage space.

In more common designs, aggregation tables are deleted after data cubes are loaded. In this design, we store the aggregation tables for a number of reasons. First of all, storing aggregation table enables fast loading to data cubes. Secondly, it makes it possible to generate temporary data cubes. Using the example in the above discussion, a user needs to make temporary queries to dimensions BCDEO. No such cube is readily available because the request for BCDEO is very rare and temporary; it is not necessary or even wise to store a cube BCDEO just for this type of queries. In order to create the necessary cube, the user can simply load the aggregated tables ABCD and EOJB from storage and create the cube from existing tables instead of regenerate corresponding aggregated tables from RDB and create cube BCDEO by going through star schema again.

Example of Multidimensional Cube

Using bill collection as example to illustrate a cube, following tables will try to show a three dimension cubes.

The central fact table of this cube is the bill collection status, i.e, how many people have paid their bill in time, and how many people are delinquent. The dimensions of this cube are Customer Age Ranges, Service Types, and Customer Income Level. Note that, Customer Age Ranges and Customer Income Level are both part of the customer information in the relational DB and star schema. Here, since end users are interested in studying if the bill payment behaviors of customers are related to their age and income level; it is, therefore, necessary to take these two fields right out from the customer information from the star schema and use them as individual dimensions.

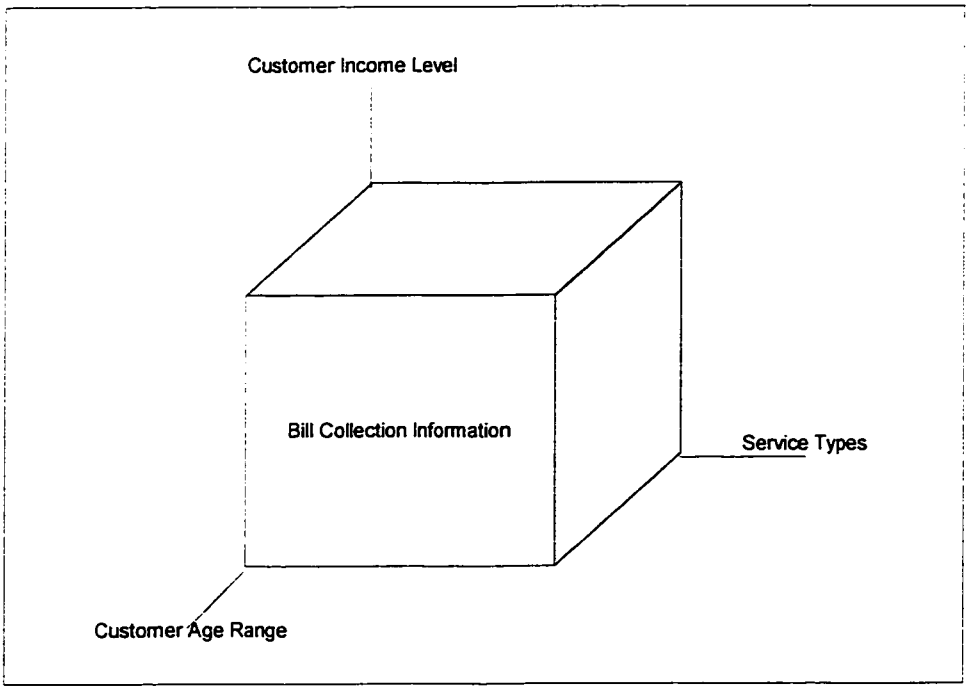

Figure 4.8: Bill Collection cube.

Originally, this block should have one more dimension, the time dimension. Since it is difficult to draw a four-dimension-block, the time dimension of a cube is normally viewed through a series of cubes at different time period. In the above example, the time dimension is removed.

Before showing any example of multidimensional data, the following table consists of measures of each dimensions of this cube. In this example, age range of the customer is just one of many measures. Other measures are customer location, customer occupation, customer credit history, customer payment methods, etc. The customer age range is only shown as an example to illustrate the way multidimensional data is stored in a cube. Similar conditions apply to both service types and customer income levels. There are more service types and income levels listed in the actual bill collection cube than in this example.

Table4.5: Dimension Measures of Bill Collection Cube.

Cus. Age Range	**Service Types**	**Cus. Income Level**	**Bill Payment Status**	
20 to 25	Regular	>2.000	Paid	
25 to 30	GPRS	2,000 – 5, 000	Delinquent	
30 to 35	Short Message	5,000 – 10, 000		
…	….	…		

When end user drilling down to detail, the table will be the same as in table 4.6:

Table 4.6: Multidimensional data in Bill Collection Cube.

Cus. Age Range	**Service Types**	**Cus. Income Level**	**Bill Payment Status**	
20 to 25	Regular	>2,000	Paid	5904
20 to 25	Regular	>2,000	Delinquent	2578
20 to 25	Regular	2,000 – 5,000	Paid	27463
20 to 25	Regular	2,000 – 5,000	Delinquent	3758
20 to 25	Regular	5,000 – 10,000	Paid	1746
20 to 25	Regular	5,000 – 10,000	Delinquent	837
20 to 25	Regular	….	….	….
20 to 25	GPRS	>2,000	Paid	1394
20 to 25	GPRS	>2,000	Delinquent	264
20 to 25	GPRS	2,000 – 5,000	Paid	2748
20 to 25	GPRS	2,000 – 5,000	Delinquent	593
20 to 25	GPRS	5,000 – 10,000	Paid	738
20 to 25	GPRS	5,000 – 10,000	Delinquent	109
20 to 25	GPRS	…	…	…
20 to 25	Short Message	>2,000	Paid	2274
20 to 25	Short Message	>2,000	Delinquent	276

20 to 25	Short Message	....	....	....
20 to 25	....	....	....	....
25 to 30	Regular	>2,000	Paid	2837
25 to 30	Regular	>2,000	Delinquent	638
25 to 30	Regular	2,000 – 5,000	Paid	14738
25 to 30	Regular	2,000 – 5,000	Delinquent	2849
25 to 30	Regular	5,000 – 10,000	Paid	27748
25 to 30	Regular	5,000 – 10,000	Delinquent	2920
25 to 30	Regular	...	...	...
25 to 30	GPRS	>2,000	Paid	3746
25 to 30	GPRS	>2,000	Delinquent	847
25 to 30	GPRS	2,000 – 5,000	Paid	27573
25 to 30	GPRS	2,000 – 5,000	Delinquent	1835
25 to 30	GPRS	....	....	....
25 to 30	Short Message	....	....	....
25 to 30	....	....	....	....
30 to 35	Regular	>2,000	Paid	....
30 to 35	Regular	....	....	....
30 to 35	....	....	....	....

The numbers in the last column are the total case number satisfies the condition given, or listed as dimensions. For example, there are total of 5904 people who are between ages of 20 to 25, with less than RMB2, 000 monthly incomes, currently subscribing the regular service and paid their bills on time. At the same time, there are total of 2578 people who are between ages of 20 to 25, with less than RMB2, 000 monthly incomes, currently subscribing the regular service and are late in bill payment. Also, based on above results, the end users are able to tell which demographic group shows a better bill payment habit. This can be obtained by a simple calculation and comparison. In group 1, in which people are between age 20 to 25, with less than RMB 2, 000 monthly incomes, currently subscribing the regular service, there are 2578/5904 = 0.467, or 46.7% of people pay their bills late. In group 2, in which people are between 25 to 30, with less than RMB2, 000 monthly incomes, currently subscribing the regular service, there are 638/2837 = 0.225, or 22.5% of people pay their bills late. These calculations can go on with different demographic groups, or income levels. The comparison

result will then tell if payment habits are related with age and, or income level. Based on above calculations, the end user may have a hypothesis that people who are older are more responsible in their bill payments. And, if this hypothesis is true, then the enterprise is able to use the result in their decision making. However, with just two calculations and one comparison are not sufficient to conclude the truthfulness of the hypothesis. It will need massive amount of data to support large number of calculations and comparisons. In order to prove a hypothesis, it usually takes at least 2 years of data. One thing has to be mentioned here is that all above calculation and comparisons can now be done by various front end tools, such as query, reporting and OLAP tools.

As one can see from above table, table 4.6, there is a lot of redundancy in data stored in the Bill Collection cube. Considering that this example is only a three dimensional cube, it is logical to deduce that the redundancy issue would deteriorate rapidly when the number of dimensions increases.

Limitations of the revised design

The revised design, though better than its predecessor, still has limitations. The first limitation is from the pre-decided hardware and software. B Mobile Communication Co. Ltd has pre-decided to use one "shark" data storage device with 5TB storage space, and two IBM RS/6000 S85s. One runs IBM's DB2, and one runs IBM's MOLAP server. Deciding and buying the hardware and software system prior to any careful studies of the project limits the choices for the designers, resulting in reduced performance and functionality. In this particular case, the ER Model storage requirement is around 2TB. The total storage capacity is around 5TB. This leaves little room for expansions or unexpected surge in data inflow.

4.4 Recommended Design

Like many other IT fields, there is no panacea for all data warehousing and data mining projects. Based on the current market situation dominated by two major players, we feel that the following two possible configurations of hardware and software are best suited to implement the hybrid architecture we discussed above. The choices are made to maximum protect the system resources of the client and facilitate development and application.

Solution 1 (IBM based)

IBM is a very important vendor in the field of data warehouse that provides both hardware and software products. In data warehouse market, IBM has its edge. Because of its dominance in the mainframe market, a large amount of data in the industry, which will potentially be placed in data warehouses, is stored and managed on IBM computers. These companies have always been IBM clients and been using IBM system software and management tools. It makes it easier to perform data extraction and data transformation when setting up and using IBM data warehouse. IBM Solution hardware configuration is presented in figure 4.8, and explained below.

Data warehouse server uses two multiprocessor UNIX servers. Alternatively, it can also use one multiprocessor UNIX server with multiple partitions. The two UNIX server or partitions on one UNIX server will install and run DB2 EEE data warehouse manager. It can divide physical data warehouse into sections based on load-balance principle. It is also possible to configure logical parallel data warehouse servers. Currently IBM S85 has some unique advantages that make it a

likable choice. S85 has open UNIX interface. Choosing this OS is purely based on client's familiarity with the interface. IBM has many powerful OS. Some of them share the common UNIX interface. Others, such as AS/400, have their own command language and interface. Another decision helped to finalize S85 as central OS is that maintenance would be much easier because support staff are familiar with the interface.

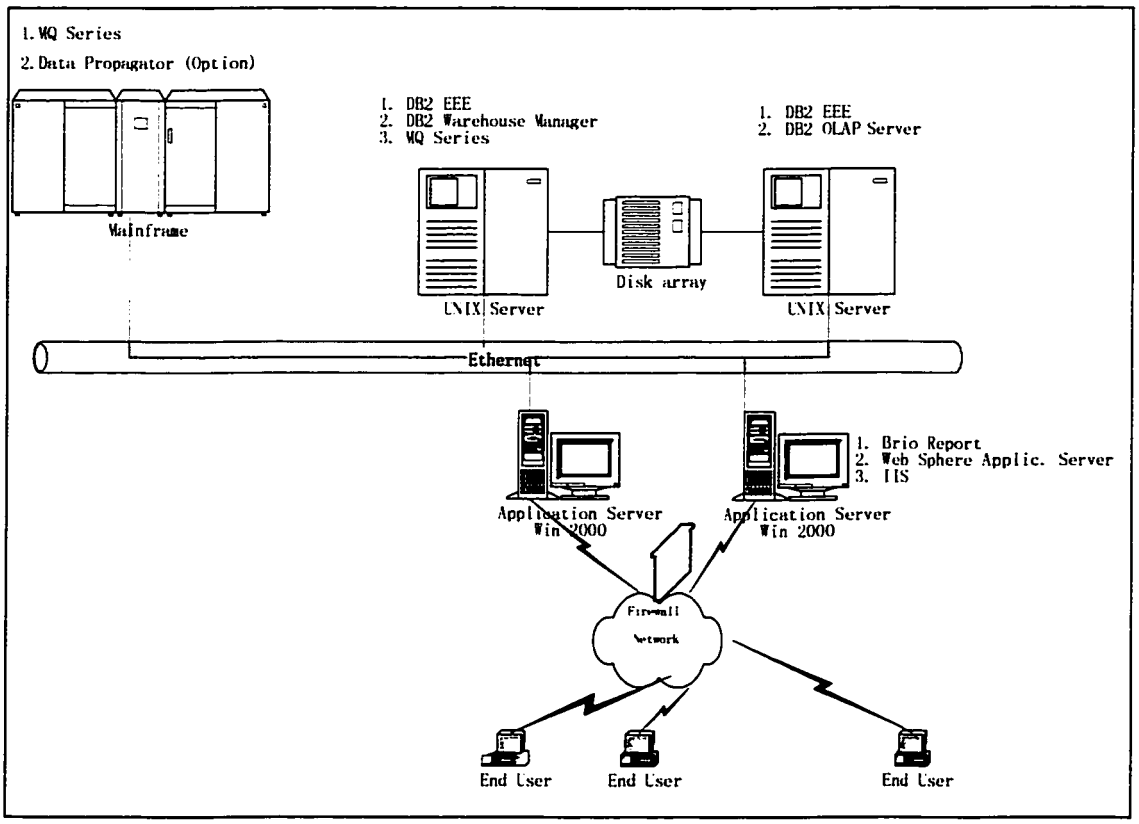

Figure 4.9: Solution 1 Hardware Configuration

One of the servers/partitions will install and run DB2 OLAP Server to function as a server for multidimensional analysis. Another server/partition will install DB2 data warehouse manager to provide ETL and atomic data administration for the entire system. It is possible to install IBM MQ series on this server based on need.

For the center server that runs core business systems, we recommend to install Data Propagator to provide dynamic data extraction, or other third party data duplication tools such as Data Mirror Transform Server and so on. It is, however, necessary to perform feasibility test and ascertain implementation support.

1. Application server will use multiple PC servers running Win 2000. Front end user applications such as Brio reporting tools can be installed on these PC servers. Java application server can use WebSphere. Web server can select IIS to support web interface for DCOM. Development for DSS can be done on application servers.
2. End user interface is web-based. Users will use browsers to access system.

- Software configuration for solution 1 is presented in figure 4.9, and explained below.

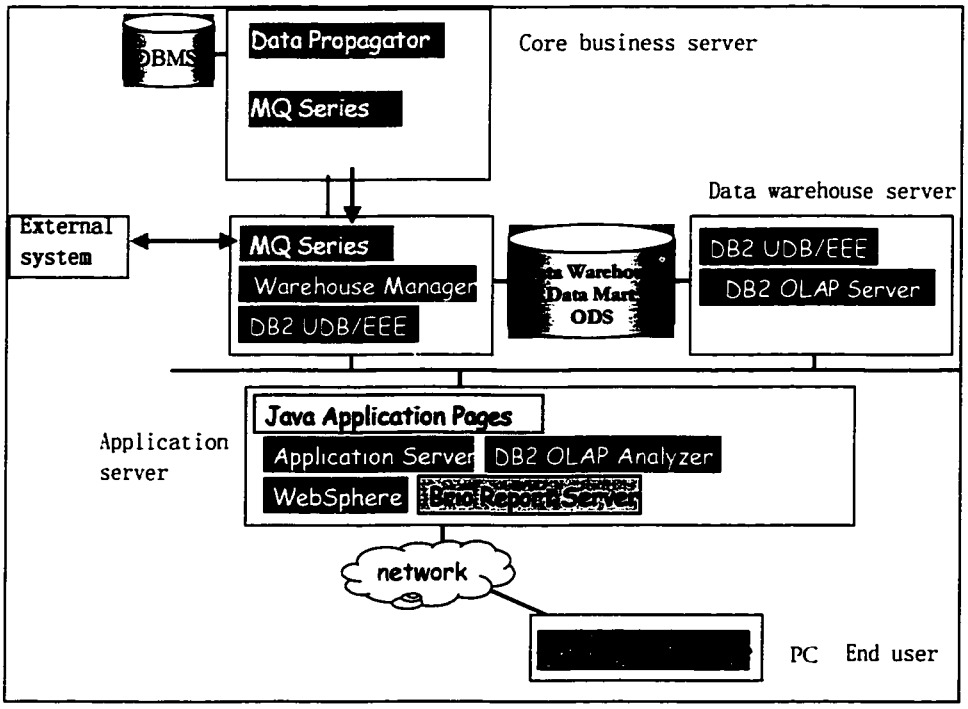

Figure 4.10: Software Configuration for Solution 1

1. Data warehouse server: DB2 UDB

Reasons for this selection: massive parallel processing structures, multidimensional cluster indexing to effectively support multidimensional analysis and access, dynamic mapping indexing to expand on data warehouse, provide PMML miner scoring mechanism directly supporting Intelligent miner scoring service, instance processing capability suitable for constructing active "data warehouse", single platform to support data warehouse, operational data and data marts. In addition, its open structure also meets demand.

DB2 Universal Database Enterprise-Extended Edition (DB2 EEE in short) version makes a better candidate than DB2 EE. DB2 EE is a fully function, web-enabled client/server RDBMS. It is intended for large and mid-size departmental servers. In addition to the functionality provided by DB2 Workgroup Edition (DB2 WE), DB2 EE also includes a DB2 Connect component that enables one to connect to host databases. DB2 EE also offers numerous extensibility features, such as extenders for spatial analysis, in-memory searches, database control for documents stored outside the files system, etc [41]. However, DB2 EE is limited by its data processing capacity. It can only process up to 500GB of data. The largest table in this project is approximately 1.87TB. Then, DB2 EE version will not be able to handle. Other than this, DB2 EE can not partition data across multiple database servers. DB2 EEE version, however, has all functionalities as DB2 EE version but without constraints on the amount of data it can process, and it provides the ability to partition the database across multiple computers (which all have to be running the same OS). In other words, the databases can grow to sizes that are limited only by the number of computers. DB2 EEE is intended for the largest (terabyte +) data warehouse and

OLAP workloads, or for high-performance online transaction processing (OLTP) requirements [41].

2. Data warehouse manager and data extraction, transformation and loading system: DB2 Warehouse manager

   Reasons for this selection: compatible with DB2 UDB. Not only provide ETL functions, it also support the entire data aggregation process from data warehouse DB2 to multidimensional data marts DB2 OLAP server. In addition, it is the atomic data administration center for the system.

   On top of DB2 Warehouse Manager, we consider Data Propagator for the server to perform online cumulated data extraction. It is used to monitor the DB2 log on the main operation server, capture data variation and transfer variations into the data warehouse system. Similar third party products include Transform Server by Data Mirror.

   MQ Series can be considered for data transfer platform.

3. Multidimensional Analysis server: DB2 OLAP server

   Reasons for this selection: compatible with DB2 UDB. It supports ROLAP and MOLAP storage format and provides seamless drill operation in between two formats. It provides excellent application development interface and interfaces with many other third party front-end tools.

4. Front-end analysis tools: Brio Report

   Reasons for this selection: it does not have multidimensional logic model management, simple and effective. It is suitable to use in conjunction with OLAP server and has good web interface access abilities.

Solution 2

Oracle is another complete data warehouse solution provider. As a leading vendor in database, it has ample experience in database management and powerful database development tools. Solution 2 hardware selection is presented in figure 4.10, and explained below.

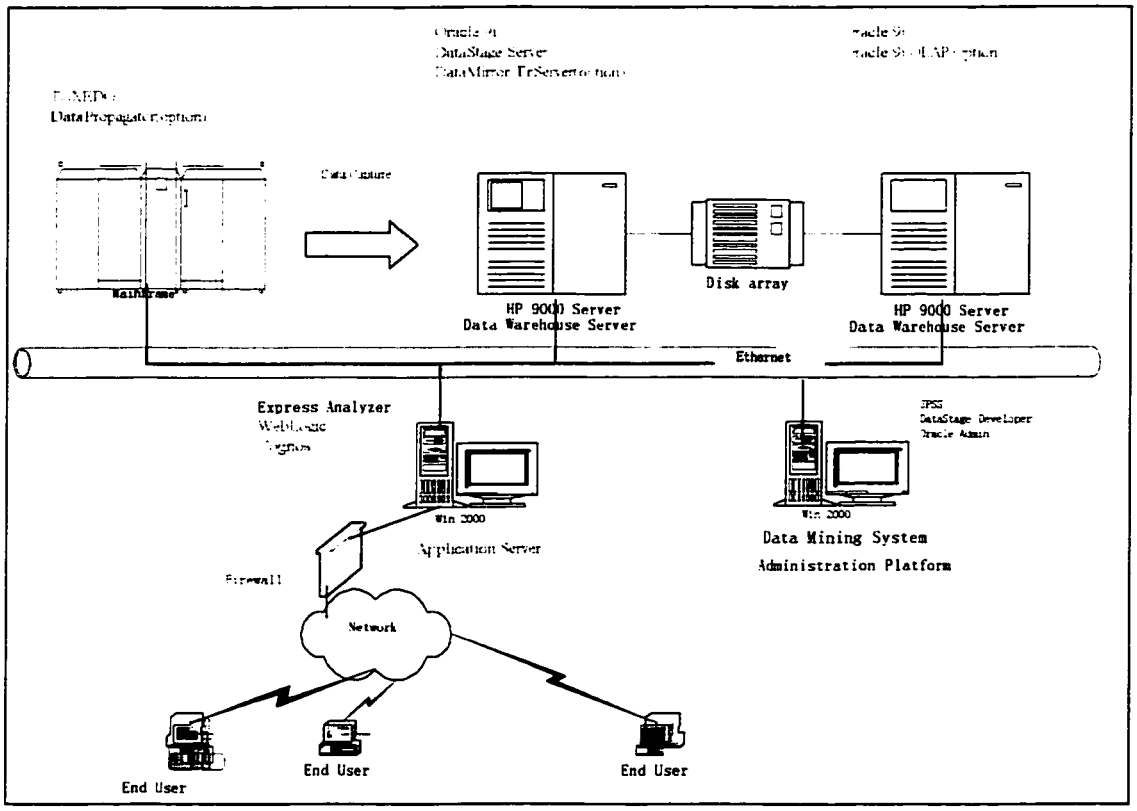

Figure 4.11: Hardware Configurations for Solution 2

1. System server uses two multiprocessor HP 9000 servers, each with Oracle 9i data warehouse server installed. OLAP tools are installed on the data warehouse servers. The system will partition physical data warehouse based on load-balance principle to perform segmentation, load balancing and error handling. It is possible to install multiple logical parallel data warehouse server (suggestion).

2. ETL environment is installed on one of the servers. Possible products include DataStage Server or Warehouse builder
3. Data mining system will use PC servers running Win 2000. Data mining tools SPSS will be installed on these PC servers.
4. System management platform will be Win 2000, running DataStage Developer/Administrator and Oracle Admin to provide system management functions.
5. End user interface is web-based. User will access the system through browsers.

The software selection of solution 2 is presented in figure 4.11, and explained below.

System software will use Oracle database software line, main components are:

1. Database/data warehouse server: Oracle 9i

   A database system that supports instance processing and decision support and includes options such as content management and other complicated data and information

2. Data warehouse manager and data ETL system: Warehouse Builder. Also can use DataStage® by Ascential Software [45]

   DataStage® is a leading data integration tool in the industry. It can manage data warehouse system using atomic data management methods, provides graphic management interface and supports XML data processing. DataStage® product family is high performing, highly scalable extraction, transformation and loading (ETL) solution with end-to-end Meta data management and data quality assurance functions. More information about Ascential software or DataStage can be found at: www.ascentialsoftware.com

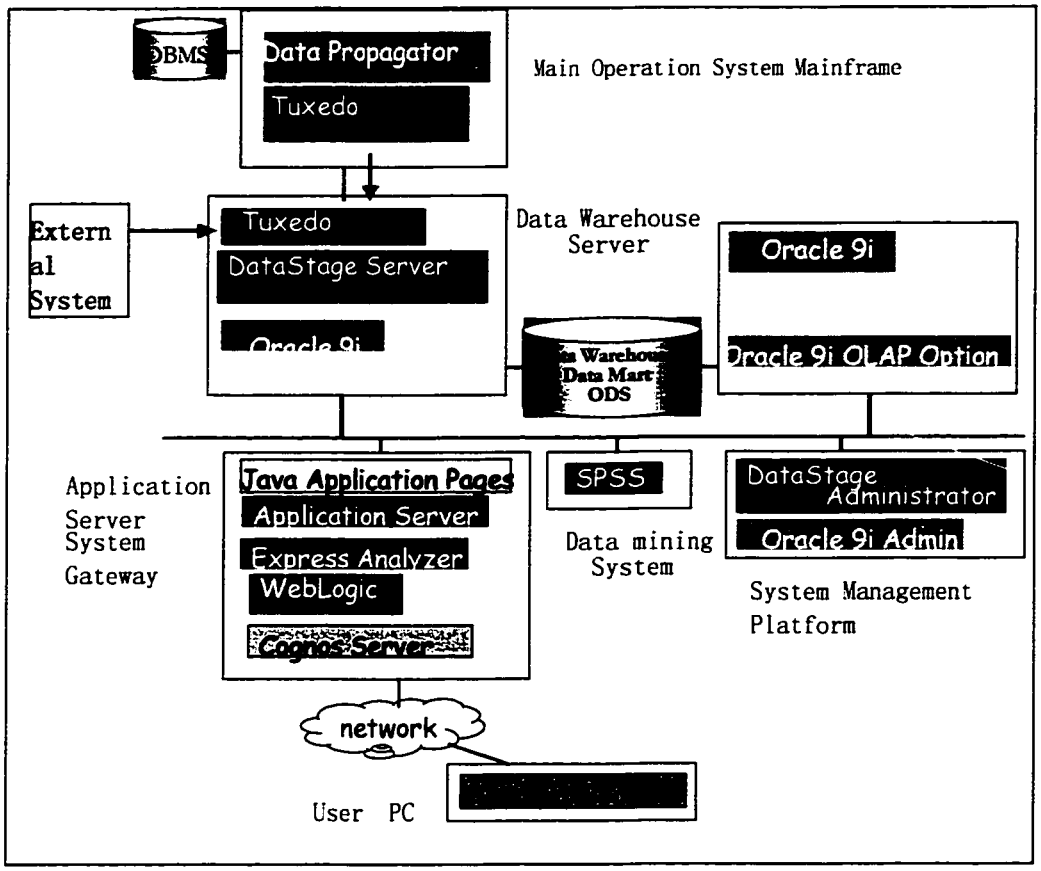

Figure 4.12 Software Configurations for Solution 2

3. Multidimensional analysis server: Oracle 9i OLAP option

   Direct OLAP solution from relational database, it represents the direction of development in this field.

4. Front-end analysis tool: Cognos

   PowerPlay by Cognos [44] is a leading front-end analysis tool. Its strength lies in fast and simple query and report functions. As with all other desktop systems, however, it has limited scalability. Recently, Cognos introduced its own multidimensional data store: PowerCube. This puts Cognos in an awkward position in its overall system. It makes Cognos a competition to other large database systems because database vendors also want to enter the data marts market. Presently, tools from Cognos will

impact on system configurations. More information about Cognos can be found at its website: www.cognos.com.

Data warehouse is a process of system integration. Its construction is not a predetermined routine. It can be constructed by different products from different vendors.

Hardware comparison

In the above two solutions, operating system of data warehouse server uses UNIX platform in both IBM RS/6000 and HP 9000. System network uses TCP/IP network. Data extraction uses end-of-day batch loading. From hardware platform and network structure point of view, both systems have strong processing capability. At the same time, both systems are stable, expandable, secure and easily managed and maintained. The main difference between the two solutions lies in the choices of software from different vendors. Therefore, we now discuss the function and characteristics of the software selections in the two solutions.

Software comparison

1. Data warehouse server

   Solution 1 (IBM Based)

   Due to its dominance in the mainframe market, a large amount of data in the industry, which will potentially be placed in data warehouses, is stored and managed on IBM computers. These companies have always been IBM clients and been using IBM system software and management tools. It makes it easier to perform data extraction and data transformation when setting up and using IBM data warehouse.

   - Main characteristics

IBM provides a complete Business Intelligence (BI) solution based on visual database. The solution is highly integrated and incorporates object-orientated SQL.

- Main tools

  Data warehouse tool Warehouse manager, OLAP tool DB2 OLAP server and data mining tool Intelligent Miner for data can be seamlessly integrated to support complex BI applications.

  Visual Warehouse (VM) by IBM is a powerful integration environment. It can be used in data warehouse modeling and atomic data administration. It can also be used for data extraction, transformation, loading and placement.

  Essbase/DB2 OLAP server supports multidimensional database. It is a HOLAP server that combines ROLAP and MOLAP. After Essbase [48] completes data loading, data will be stored in DB2 UDB database assigned by the system. For more information about Essbase, visit www.essbase.com

  QUEST [49] is a multitasking data mining system developed by IBM Almaden research center. Its objective is to perform efficient data mining in order to establish basic construct for the development and application of a new generation of Decision Support System. System provides many mining functions. The mining algorithms are applicable in databases of all sizes. Information of QUEST can be found at: www.almaden.ibm.com/cs/quest/

- Application notes

IBM does not provide complete data warehouse solution. It does, however, allow third party data warehouse tools. For example, query tools can use Business Object by Business Objects [46], statistical analysis tool can use SAS system by SAS [47]. (Business Object is an excellent front-end analysis tool. Its strength lies in its ability to directly access many relational databases. It can be integrated with MetaCube by Informix, Express by Oracle and Essbase by Arbor. Business object has strong reporting functions, easy to use and high level of localization as its advantages. As a desktop OLAP tool, however, it is limited by local multidimensional transformation base – Hyper Cube. Therefore it is limited in its processing ability of large amount of data and in its scalability. At the same time, the system does not include an automated optimization mechanism in data aggregation and sampling. www.businessobjects.com) (SAS has powerful functions of data sampling, data management, data analysis and information presentation. It is the best tool for decision support systems. www.sas.com)

Solution 2 (Oracle based)

Oracle is the market leader in database. In database market, Oracle has its unique advantage. Oracle is the first to develop multidimensional OLAP on database. A few years ago, Oracle acquired IRI, a multidimensional database supplier, introduced Express multidimensional database and provided solution to online analysis in the form of MOLAP.

- Main characteristics

Data warehouse solution by Oracle includes leading database platform, development tool and applications in the industry. It can provide a series of data warehouse tools and services. It has multi user data warehouse management ability, massive parallel processing, various partition schemes, strong interaction with OLAP tools, and fast and agile data movement mechanism.

- Main tools

  Oracle provides a series of data warehouse tools

  Oracle 9i (or 8i) is the core of data warehouse.

  Oracle Warehouse Builder includes integrated data modeling, data extraction, transformation and loading, aggregation, atomic data management, etc.

  Oracle Developer Server is enterprise level application system development tool. It supports object-orientation and multimedia. It can generate Client/Server and web-based applications. It has high development efficiency and scalability.

  Oracle Discoverer is a tool of query, reporting, drilling, rotating and web publishing for end users. It can help users to fast access relational database, which enables user to make better decision based on sufficient information.

  Oracle Darwin is a data mining tool in data warehouse. It uses simple graphic interface and supports different data mining methods such as decision-tree, neural network, etc. It also supports parallel processing of massive data. Final analysis results can be integrated into existing systems.

- Application notes

Oracle data ETL tools require SQL drafting. It has a lot of difficulties when processing complex data transformations. The front-end tools in Oracle are not easy to use. It relies heavily on third party products.

As we can see from the above discussion IBM solution and Oracle solution both have their unique advantages. The choice between these two for a specific project would largely depend on the nature of the project, the environment and requirements of the client.

More relevant to B Mobile project, the current design used in this project is based on existing hardware and software system. However, the unusually large amount of data puts extra strain on the selection of software and hardware. To maximize the performance, additional processing and storage capacity is very necessary. We will use the IBM solution platform to discuss the necessary capacity for this particular project.

Currently, the system employs two RS/6000 S85s to handle data process. Each S85 has 6 CPUs installed (S85 can have maximum 12 CPUs). According to IBM's study, a single CPU in S85 is able to handle up to 200 GB data per year. The largest fact table is 1.87 TB per year. The basic data need to be processed is around 2TB per year. The most simple calculation gives 2TB/200GB = 10. Therefore, the current system is limited by the hardware setting, which only allows for one year of data to be processed.

The total storage needed for cumulated data over a year is 3 times the amount, in this case, 1.87 TB * 3 = 5.61 TB. The reason why the overall storage space required is the data amount times 3 is because there are three major parts of data in the DB. The first part is the basic data. The second and third parts are derived from the basic data; they are multi-dimensional

data and index. After optimization, we will need at least 5TB of storage space. The current system has one IBM Enterprise Storage Server (ESS), otherwise known as Shark, of 5TB capacity. As a result, the system can only store cumulated data for one year.

The reason for building this data warehouse is to help B Mobile to perform data analysis; therefore, the more historical data this warehouse can store, the better and more accurate the analytical result would be. Usually, two or three years of data should be sufficient for an accurate analysis. There should be at least 5 TB * 3 years = 15 TB storage space available within the system. After optimization, we would still need 10 TB of storage space minimum.

To process cumulated data over a three year period, we would need 3 fully loaded S85, i.e, each with 12 CPUs. The total amount of data they can handle is 36x200GB=7.2TB. Two of them will run DB2, and the third runs MOLAP server.

Obviously any system can be made better by more capital investment. The discussion presented above is an upgrade on the already selected hardware and software. It requires some degree of budget increase for the project. One can always argue that there may be superior hardware and/or software from other vendors. When alternatives of hardware and software are taken into account, choosing a design then becomes a business decision rather than a technical one, which is beyond the scope of this thesis.

4.5 Summary

In this chapter, we reviewed and compared three designs of a data warehouse. Limited by the choice of hardware and software, the first and the revised designs are proposed as best-effort solutions to the data warehouse project of B Mobile Comm. Co.. The first design is a straightforward MOLAP solution. It has the advantage in terms of performance that is typical

for MOLAP. However, this design is inflexible to possible future modifications to the system. Secondly, the queries and reports this system can provide are rather limited comparing to the revised design. The revised design is an improvement from the first design. It includes multiple storages for data with different levels of granularity. This feature enables many queries and reports to be generated from the nearest level of granularity, thus improves performance of OLAP. In addition, it provides great flexibility and expandability for future modification or addition to the system.

Two possible implementations are presented, IBM based and Oracle based. The hardware and software for these two solutions are discussed in some detail. The comparison is also presented to show their characteristics. Based on the IBM solution, a third design for B Mobile project is presented. This solution is an ideal solution to the business requirement of the client. The main difference between this design and the second revised design is the choice of hardware. There is little change in the architecture design of the data warehouse. The third design inherits all the design features and advantages of the revised design, without the limitation on storage capacity. Due to the limitation of the predefined hardware, the data warehouse in the revised design can only store data for up to two years. This limit may affect the accuracy of analysis from a statistical standpoint. It is desirable for this data warehouse to be able to store data for up to five years. The most suitable hardware choice is presented in the third design. This point, of course, becomes an issue of investment and return, which is beyond the scope of our discussion.

Chapter 5

Conclusion

In this thesis, we have reviewed briefly the concept of data warehouse, common elements of data warehouse and some key technical issues in data warehouse. We have come to the following conclusions.

Data model design in a data warehouse project is not a fixed, standardized procedure, and there are no rules available on where to use what types of model. It is really decided by the requirements. Current disagreements on this subject between relational data model and multidimensional data model are really not necessary. Each side has its own strength and weaknesses. It is unwise to rush a choice of data model before understanding the context and objectives of the data warehouse. In stead, the choice of data model should be subject and based on the project itself. There are some articles [32, 34] shared the same view.

It is generally not necessary to include ER model, start schema and the aggregated tables in one data warehouse project. However, after closely studying the requirements and business objectives of the client in this project, we found that the final cubes would not be able to satisfy all needs. In stead, certain kinds of analysis are still based on the smallest granularity even down to a single client. Such functions can only be completed by client-end applications such as Brio. In this scenario, ER model and star schema can not be avoided. It is mainly for this reason that they are implemented in this design.

Generally speaking, the architecture of a project really depends on the requirements and future business development of the client. If the client has clearly defined objectives from a data warehouse, understood the data processing steps, defined the smallest granularity for analyzing data, and most importantly all possible future development of the business, the structure of the entire data warehouse can be easily defined and the architecture can be designed. On the other hand, if the client is rather unclear about what he wants or where the business is heading, it is then unwise to lose any data while implementing the data warehouse. In this case, it is better to keep the data in their atomic forms in a scaled database for later use. The architecture design of this type of projects should always leave space for future changes. In these cases, transformation stages such as star schema and aggregate tables are very useful.

Secondly, there is the issue of "real-time" between the two OLAP methods. ROLAP supporters argue that since MDB is updated by batch consolidation process at predetermined time only, MOLAP users are not able to access the "real-time data". Whereas in ROLAP, end user can directly access the Relational database when there is need for "real time" data [11]. On the other hand, MOLAP designers argue that Relational database itself is derived from operational data through ETL processes at predetermined time interval and can not qualify as true "real time" [12]. While both sides provide valid argument on this particular issue, it is not the main concern of the project we have at hand. In this project, both RDB and MDB are utilized at different stages of the data flow. MDB provides direct access to predefined multidimensional data cubes to the majority of the end users. For other uses, when such predefined aggregated summary data does not meet their requirements, they have the option of constructing individual SQL query routines to access the Relational Database to generate summary data tailored towards their needs. In both scenarios, the need for true "real time"

data is minimal since the purpose of OLAP is to support decision making based on historical analysis of business data.

Finally, the choice between MOLAP and ROLAP is project dependent. Similar to the discussion above regarding the design of overall data warehouse architecture, when the objectives of the client is clearly defined, the functional requirements of the data warehouse is determined, and the amount of data is relatively small, MOLAP would likely be a better choice because of its superior performance and intuitive querying mechanism. On the other hand, ROLAP would become a more suitable candidate for OLAP when the amount of data is exceedingly large, and the client does not want to leave out anything. This is because ROLAP has much better storage capacity, which makes it possible to store large amount aggregated information. It is then possible to construct very complicated and elaborate queries to best fit special needs from individual users.

Contribution:

The core of the thesis is the B Mobile project. Many people participated in this project. My exact involvement is the following. First, I was responsible for analyzing client requirements. Based on the requirement analysis results, I proposed designs of the cubes in MOLAP. I was also responsible for the final verification of the cube design together with the implementation group. In addition, I also participated in overall architecture design and star schema design.

BIBLIOGRAPHY

[1] Devlin, Barry, *"Data Warehouse – from Architecture to Implementation"*, Addison Wesley Longman, Reading, Mass. 1997

[2] Inmon, W.H., *"Building the Data Warehouse"*, 2nd ed., John Wiley & Sons, New York, 1996

[3] Kimball, Ralph. *"The Data Warehouse Toolkit"*, John Wiley & Sons, New York, 1996

[4] Ralph Kimball, Laura Reeves, Margy Ross, Warren Thornthwaite, *"The Data Warehouse Lifecycle Toolkit – expert methods for designing, developing, and deploying data warehouses"*, John Wiley and Sons, 1998

[5] Sam Anahory, Dennis Murray, *"Data Warehousing in the real world"*, Addison Wesley, 1997

[6] T. Barclay, R. Barnes, J. Gray, P.Sundaresan, *"Loading Databases Using Dataflow parallelism"*, SIGMOD Record, 23(4), December 1994

[7] Ming-Chuan, Wu, Alejandro P. Buchmann, *"Research Issues in Data Warehousing"*, BTW 1997: 61-82

[8] Colin J. White, *"An Analysis-led Approach to Data Warehouse Design and Development"*, Database Associates, January 2000, Version 1

[9] Efrem G. Mallach, *Decision Support and Data Warehouse Systems*, Irwin McGraw-Hill, 2000

[10] OLAP Council, The. *"Guide to OLAP Terminology."* January 1995, Available on-line: http://www.olapcouncil.org/whtpap.html

[11] *"The Case for Relational OLAP"*, MicroStrategy White Paper, 1995

[12] *"Analytical Processing: A comparison of Multidimensional and SQL-based approaches"*, Hyperion White paper, 2000, http://www.essbase.com/download_files/resource_library/white_papers/analytical_processing.pdf

[13] *"Next-Generation Telco-class Business Operating Support System"*, China Telecommunications Construction, V 13, July 2001

[14] Codd, E.F., S.B. Codd, C.T.Sally, "*Providing OLAP (On-Line Analytical Processing) to User Analyst: An IT Mandate.*" http://www.arborsoft.com/OLAP.html

[15] http://pwp.starnetinc.com/larryg/articles.html

[16] http://www.brio.com

[17] E. Thomsen. "*OLAP Solutions: Building Multidimensional Information Systems.*", Wiley, 1997.

[18] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson., "*Extending Practical Pre-Aggregation in On-Line Analytical Processing*". , In *Proc. of VLDB*, pp. 663–674, 1999.

[19] R. Winter. Databases: Back in the OLAP game. *Intelligent Enterprise Magazine*, 1(4):60–64, 1998.

[20] T. Zurek and M. Sinnwell., "*Data Warehousing Has More Colours Than Just Black and White.*", In *Proc. of VLDB*, pp. 726–729, 1999.

[21] N. Raden, "*Modeling the Data Warehouse*", White Paper, from http://www.netcom.com/.

[22] *"The Role of the Multidimensional Database in a Data warehousing Solution"*, White Paper, 1995

[23] K.Sahin, *"Multidimensional Database Technology and Data Warehousing"*, Database Journal, December 1995

[24] M.J.Saylor, M.G. Acharya, R.G. Trenkamp, *"True Relational OLAP: The Future of Decision Support"*, Database Journal, November, 1995

[25] R. Findelstein, *"Understanding the Need for Online Analytical Servers"*, White Paper, from http://www.arborsoft.com/papers/

[26] C.J. Date, "*An introduction to database systems*", Vol. I, Addison Wesley, 1990.

[27] Ralph Kimball, "*A Dimensional Modeling Manifesto*", DBMS Magazine, (Vol 10, Num 9) Page 59, August, 1997

[28] W.H.Inmon, "*The Problem with Dimensional Modeling*", Available from: http://www.billinmon.com/library/library_frame.html

[29] The OLAP Report, "*What is OLAP?*", http://www.olapreport.com/fasmi.htm, February 19, 1998

[30] C. Ballard, D. Herreman, D. Schau, R. Bell, E. S. Kim, A. Valencic, "*Data Modeling Techniques for Data Warehousing*", White Paper, IBM, February, 1998

[31] Colin White, "*Data Warehousing: Cleaning and Transforming Data*", White Paper, EvalTech, Available: http://www.evaltech.com/wpapers/dwcleansing.pdf

[32] Colin J. White, "*Building a Corporate Information System: The Role of the Data Mart*", Database Associates International, Inc., February, 1997

[33] Joseph M. Firestone, "*Dimensional Modeling and E-R Modeling In the Data Warehouse*", White Paper, Executive Information Systems, Inc., June 22, 1998

[34] Susan Gallas, "*Kimball vs. Inmon*", DM Direct, Available from: http://www.dwinfocenter.org/articles.html , September, 1999

[35] Sun Microsystems, "*Data Warehousing - Scaling the Data Refinement Process*", DM Review, October 6, 2001

[36] T. Barclay, R. Barnes, J. Gray, P. Sundaresan, "*Loading Databases Using Dataflow Parralism*", SIGMOD Record 23(4): 72-83, 1994

[37] Frank Teklitz, "*The Simplification of Data Warehouse Design*", White Paper, Sybase, 2000

[38] Thomas J. Kelly, "*Dimensional Data Modeling*", White Paper, Sybase, 1998

[39] Ralph Kimball, "*Is ER Modeling Hazardous to DSS?*", DBMS Magazine, (Vol 8, Num 11) Page 17, October, 1995

[40] D. Dodds, H. Hasan, E. Gould, "*Relational versus multidimensional databases as a foundation for online analytical processing: Lessons from two case studies*", Available http://iris22.it.jyu.fi/iris22/pub/Gould_IRIS22_MDDB3.pdf

[41] P. Zikopoulos, R. Melnyk, "*Which Distributed DB2 Edition is Right for you?*", IBM on-line library, July 2002, available: http://www7b.software.ibm.com/dmdd/library/techarticle/0207melnyk/0207melnyk.html#section6

[42] S. Gatziu, A. Vavouras, "*Data Warehousing: Concepts and Mechanisms*", Informatik, January 1999

[43] CIOL Special Report, "*Enterprise Special: Data Warehousing*", Available at: http://www.ciol.com/content/e_ent/data_wa

[44] PowerPlay, www.cognos.com

[45] DataStage, www.ascentialsoftware.com

[46] Business Objects, www.businessobjects.com

[47] SAS, www.sas.com

[48] Essbase, www.essbase.com

[49] QUEST, www.almaden.ibm.com/cs/quest/