BUILDING A DATA MART USING STAR SCHEMA

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Gauree Bhole
Summer 2010
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Gauree Bhole:

Building a Data Mart Using Star Schema

Carl Eckberg, Chair
Department of Computer Science

Robert Edwards
Department of Computer Science

J. Carmelo Interlando
Department of Mathematics and Statistics

May 15, 2010
Approval Date
Copyright © 2010
by
Gauree Bhole
All Rights Reserved
DEDICATION

I dedicate my work to my husband Kedar Bhole, who was always there to support me in all difficult times throughout my Masters degree. I would also like to thank my parents, Mr. S. K. Deshmukh and Mrs. Sujata Deshmukh, for their continual affection and prayers all throughout my academic life.
ABSTRACT OF THE THESIS

Building a Data Mart Using Star Schema
by
Gauree Bhole
Master of Science in Computer Science
San Diego State University, 2010

Data analysis and integration are the main objectives of any database based project for a particular organization, and there are several ways described by field experts to achieve this objective.

Amongst various approaches to data base management, Data Warehousing strategies are preferred over other methods when it comes to business data. Herein, data related to specific departments or entities is extracted from operational system, processed, put into the specialized relational databases, organized and presented as actionable information. A Data Warehouse is most often based on a decisional-architecture platform, which can be used by analytical and reporting tools, serve as a digital dashboard to company executives, and provide analysis with easy-to-access data irrespective of source system. It also improves data accuracy and keeps a track of historical data.

A Data Mart is essentially a subset of a Data Warehouse. It is a flexible set of data, ideally based on the most atomic (granular) data possible to extract from an operational source, presented in a symmetric (dimensional) model, and most resilient when faced with unexpected user queries. In its most simplistic form, a Data Mart represents data from a single business process or function. The process, from its design till commissioning, entails low initial investment.

The aim of this thesis work is to design a sales-function Data Mart, using star schema strategy. It involves collecting requirements from the sales department of a given organization, and discussing a flexible Data Mart architecture to implement those requirements and to facilitate its seamless integration with Data Marts representing other business processes of the same organization.

The thesis builds the prototype of the sales data mart as applicable to a cellular telephone manufacturing company and discusses the overall architecture at an enterprise level data warehousing, including business requirements, data mart architecture, and logical and physical model design. Further, this thesis follows an “incremental approach” for building the data mart.
TABLE OF CONTENTS

ABSTRACT ...............................................................................................................................v
LIST OF TABLES .....................................................................................................................x
LIST OF FIGURES ................................................................................................................. xi
ACKNOWLEDGEMENTS .................................................................................................... xii
CHAPTER

1 INTRODUCTION .............................................................................................................1
  1.1 Why Build a Data Warehouse? ..............................................................................2
  1.2 Limitations of E-R Based Database Maintenance System ................................2
  1.3 Advantages of Building a Data Warehouse .......................................................3
  1.4 How is the Data Warehouse ‘Data’ Different? ..................................................4
  1.5 Differences Between Enterprise Data Warehouse and a Data Mart ...............4
  1.6 Building a Data Mart Incrementally .................................................................5
  1.7 Project Background ............................................................................................6
  1.8 Project Goal .......................................................................................................7
    1.8.1 Sales Information Repository (SIR) Goals ...............................................7
    1.8.2 Project Scope ............................................................................................8
    1.8.3 Anatomy of a Sale .....................................................................................8
  1.9 Project Approach ...............................................................................................9
  1.10 Common Pitfalls to Avoid While Building a Data Mart ...............................10

2 DATA WAREHOUSE CONCEPTS AND STRATEGIES ..............................................12
  2.1 Two Main Paradigms in the Field of Data Warehousing .............................12
    2.1.1 Bill Inmon’s Paradigm: Third Normal Form Data Model ....................12
    2.1.2 Ralph Kimball’s Paradigm: Dimensional Data Model .........................13
  2.2 Why Use Star Schema to Build the SIR? .......................................................15
  2.3 Summary ..........................................................................................................15

3 DATA WAREHOUSE AND DATA MART ARCHITECTURE STRATEGY ..........16
  3.1 Top Level Bus Architecture and Matrix Solution .........................................16
4.4.1 Entity Definition .....................................................................................37
4.4.2 Entity Naming .........................................................................................38
4.4.3 Attribute Naming and Organization Format ...........................................38
4.4.4 Attribute Definition .................................................................................38
4.4.5 Data Naming Abbreviations .................................................................38

4.5 Physical Model .........................................................................................39
4.5.1 Table Definition Guidelines and Table Naming ...................................39
4.5.2 Column Naming ......................................................................................39
4.5.3 Primary Key Constraint Naming .............................................................39
4.5.4 Trigger Naming .......................................................................................39
4.5.5 Aggregation Strategy ..............................................................................40
4.5.6 Indexing Strategy ....................................................................................41
4.5.7 Strategy for Handling Slowly Changing Dimensions.............................41
   4.5.7.1 Type One: Overwrite Old Data with New Data .............................41
   4.5.7.2 Type Two: Add a New Record at the Time of the Change ...........42
   4.5.7.3 Type Three: Track Changes Using a Separate Column .................42
   4.5.7.4 Type Four: Hybrid Techniques ......................................................42

4.6 Summary .....................................................................................................43

5 DATABASE DESIGN, IMPLEMENTATION AND ACCEPTANCE TESTING .................................................................44
5.1 SQL Server Database Configurations .......................................................44
5.2 Database Backup ........................................................................................46
5.3 Role Privileges ..........................................................................................46
5.4 User Privileges ........................................................................................47
5.5 Tables and Surrogate Keys, Indices ..........................................................49
   5.5.1 Data Staging Plan ..............................................................................50
   5.5.2 Surrogate Keys ..................................................................................50
5.6. The Sales Force Fact Table .....................................................................51
   5.6.1 Degenerate Dimension in Fact Table ...............................................51
   5.6.2 Incremental Loading into Sales Fact Table ........................................52
5.7 Partitioning ...............................................................................................52
5.8 Synonyms ..................................................................................................53
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Differences Between Operational and Data Warehouse ‘Data’</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Comparison of Characteristics between Dimensional Data Modeling and 3rd Normal Form Modeling</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Sample Data Warehouse Bus Matrix for CalQ Inc.</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>Budget Allocation Guidelines</td>
<td>22</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison of Relational and Dimensional Data Modeling</td>
<td>34</td>
</tr>
<tr>
<td>5.1</td>
<td>Database Roles and Their Respective Permissions</td>
<td>48</td>
</tr>
<tr>
<td>5.2</td>
<td>Types of Users and Schema Owned by Them</td>
<td>48</td>
</tr>
<tr>
<td>5.3</td>
<td>Data Tables and Their Respective Natural and Surrogate Keys</td>
<td>49</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

PAGE

Figure 2.1. Bill Inmon’s third normal form model .................................................................13
Figure 2.2. Ralph Kimball’s star schema sample model. ........................................................14
Figure 3.1. Sharing dimension – data warehouse bus..............................................................17
Figure 3.2. Sales data mart architecture................................................................................23
Figure 3.3. Data mart life cycle. ............................................................................................26
Figure 3.4. Data flow in SIR .................................................................................................27
Figure 4.1. Multidimensional cube structure. .......................................................................33
Figure 4.2. Logical view of a data model. .............................................................................37
Figure 5.1. SQL Server 2005 Configuration Manager ..........................................................45
Figure 5.2. Creation of database role and privileges using SQL Server Management Studio 2005. .................................................................................................................47
Figure 5.3. Creating s synonym using SQL Server Management Studio 2005. ...............54
Figure 5.4. Reporting tools comparison ................................................................................56
Figure B.1. Physical model for sales data mart representing star schema.............................66
Figure B.2. Assigning privileges to respective roles using Microsoft SQL Server Management Studio Express..........................................................70
Figure B.3. Scheduling job using Microsoft SQL Server Management Studio Express....71
ACKNOWLEDGEMENTS

First of all, I would like to express my deepest thanks to Dr. Eckberg, for chairing my committee and for giving me an opportunity to work on this project. Every suggestion made by him encouraged me towards development of this project.

I would like to express my heartfelt thanks to my friend Mr. Prashant Kher, a professional data warehouse analyst, designer and developer, for his continual guidance, support and motivation throughout the project work despite his busy work schedule. I am grateful to Dr. Robert Edwards and Professor J. Carmelo Interlando for being on my committee and for their help and cooperation.

I would also like to thank Mr. Mike Li, who volunteered his valuable time for supporting and guiding me through the steps of implementation of this project.

I would especially like to thank Dr. Sunil Kumar, Director of Multimedia & Wireless Networks Research group of the Department of Electrical and Computer Engineering at SDSU, and Dr. Rajni Garg, whose guidance was very rewarding throughout pursuit of my degree.

Finally, I would like to convey my gratitude to all the people, especially my father-in-law Mr. Indranil Bhole, Mr. Ishan Bhole, Mrs. Maitreyi Tapaswi, Mrs. Vaidehi Puntambekar, Mr. Venkatesh Rangarajan, Mr. Vaibhav Singhal, and my friend, Di Huang, who have supported and encouraged me in completing this thesis project.
CHAPTER 1

INTRODUCTION

In this age of information technology, information is one of the most important assets of any business or organization. The trend towards speedy information gathering and data growth is more obvious in the multinational and fast growing technological companies, including mobile manufacturing companies and telecom companies. Effectively managing the constantly changing data and processing business information has always been a big challenge. This is especially true for advanced computer technologies in order to deliver best-of-the-best business intelligence solutions to serve the needs of the industry.

Before the advent of data warehousing, techniques such as decision support system (DSS) allowed managers, supervisors and other business people to see clipboards with considerable information of certain kinds. However, gathering information from two or more different business areas and figuring out how to effectively warehouse (store) the data of an enterprise in an easily retrievable form was still a challenge. In the 1990s, William Inmon and Ralph Kimball created and documented concepts and principles behind data warehousing, which are collectively known as ‘Data Warehouse Philosophies’, which gave a lot more flexibility for handling the data. This made it very popular amongst enterprise organizations which sought a competitive advantage by gathering strategic information quickly and easily [1-3].

A data warehouse (DW) has been elaborately defined by many researchers and inventors. To illustrate with one such definition: “A data warehouse is subject oriented, integrated, time-temporal and nonvolatile collection of data specially structured for query and analysis that supports managerial decision making” [4,5]. Although this definition makes a data warehouse sound very efficient and capable, the returns from a practical data warehousing implementation have always been a divisive subject in the business world. Some organizations claim more than 100% return on investment (ROI) after building a data warehouse, while some appear to fail completely in implementing a similar project. The reason behind getting these varied ROI lies in choosing the ‘best fit approach’ for that
specific company’s or organization’s architecture. This includes dealing with internal political issues that will invariably arise during implementation. This is the most challenging aspect of data warehousing.

A true data warehouse may or may not be the correct solution for any particular organization. A large data warehouse effort can be a very expensive and risky venture. However, by prioritizing goals and solidly planning prior to spending large amounts, one can eliminate much of the risk, while rallying the support needed to succeed. Business goals and project plans are helping achieve milestones in determining the correct solution, and taking an “incremental approach” ensures that each small project phase will be more successful than the last.

1.1 WHY BUILD A DATA WAREHOUSE?

As we have seen in the introduction, one of the main reasons for creating and maintaining databases is to support business decisions by producing analytical and strategic reports from them. Entity relationship (E-R) model is a very general way of managing and maintaining data in a typical organization or company. E-R is based on the ‘relational database’ methodology. This methodology has a single data store which is most commonly called an ‘operational system’ (where the data is put in). A specific query needs to be written to get data out of this system. Applications using E-R based methodology process data directly from the operational system. For example, OLTP (On-Line Transactional Processing): where each transaction requires a single record in the database to be located, updated, and/or one or more new records to be added. Besides, these kinds of queries put an additional load on the hardware and the network supporting system, when they involve large tables with millions of rows.

1.2 LIMITATIONS OF E-R BASED DATABASE MAINTENANCE SYSTEM

Some of the limitations of using this system are noted:

- Slow retrieval of data for complex and large reports, due to, among other reasons, the unbounded number of concurrent users of an OLTP.
- No easy way to maintain a history of the specific record.
- Less flexibility for remodeling activities to accommodate new application and business areas.
• Not effective for decision support.
• Direct use of operational data, which frequently holds duplicate information, could lead to inaccuracy in analysis and reports, at times.
• Not very flexible for ‘multidimensional’ databases.

1.3 ADVANTAGES OF BUILDING A DATA WAREHOUSE

In a data warehouse or data mart strategy, more importance is given to processing and filtering the data. Herein, data related to specific department or entity is extracted from the operational system, processed, and then put into the ‘specialized relational database’ (for Extract-Transform-Load process, see Chapter 3). Hence, there is less possibility of duplicated and/or inaccurate data contaminating analytical reports generated from such a database. Basically, data is getting better organized in this methodology, and, as said by experts, data comes to have “value” to end-users only when it is organized and presented as actionable information [6-8].

In addition, several specific technologies in data warehousing can be used to dynamically improve performance of the decision support queries; for example, system design techniques like ‘star’ and ‘snowflake’ schemas (see Chapter 2). In the case of very large data warehouses, parallel processor technology can (for some kinds of particularly expensive queries) make the difference, between being able to execute decision support queries in a reasonable amount of time, and having absolutely no ability to do so.

Building a data warehouse has several advantages when it comes to a large and constantly growing data pool. Important characteristics which make data warehousing perform better than E-R are as follows:

1. Subject-Oriented: Information is presented according to specific subjects or areas of interest, and not simply as computer files. Data is manipulated to provide information about a particular subject.

2. Integrated: It is a single source of information for, and about, understanding multiple areas of interest. The data warehouse provides a one-stop shop, containing information about a product and about where it is getting sold.

3. Non-Volatile: It contains stable information that does not change each time an operational process is executed. Information is consistent regardless of when the warehouse is accessed.

4. Time-Variant: It contains the history of a subject, as well as its current information. Historical information is an important component of a data warehouse or data mart.
5. Accessible: The primary purpose of a data warehouse is to provide readily accessible information to end-users.

6. Process-Oriented: It is important to view data warehousing as a process for delivery of information. The maintenance of a data warehouse is an ongoing and iterative process.

1.4 HOW IS THE DATA WAREHOUSE ‘DATA’ DIFFERENT?

The data warehouse is distinctly different from the operational data used and maintained by day-to-day operational systems. Data warehousing is not simply an “access wrapper” for operational data, where data is simply “dumped” into tables for direct access. Table 1.1 highlights the differences between operational and data warehouse data.

Table 1.1. Differences Between Operational and Data Warehouse ‘Data’

<table>
<thead>
<tr>
<th>Differences</th>
<th>Operational Data</th>
<th>DW Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Application-oriented</td>
<td>Subject-oriented</td>
</tr>
<tr>
<td>2</td>
<td>Detailed</td>
<td>Summarized, or else refined</td>
</tr>
<tr>
<td>3</td>
<td>Accurate, as of the moment of access</td>
<td>Represents values over time, snapshots</td>
</tr>
<tr>
<td>4</td>
<td>Serves the clerical community</td>
<td>Serves the managerial community</td>
</tr>
<tr>
<td>5</td>
<td>Requirements for processing understood before initial development</td>
<td>Requirements for processing not completely understood before development</td>
</tr>
<tr>
<td>6</td>
<td>Compatible with the Software Development Life Cycle</td>
<td>Completely different life cycle</td>
</tr>
<tr>
<td>7</td>
<td>Performance sensitive (immediate response required when entering an update transaction)</td>
<td>Performance relaxed (immediacy not required, because the focus is on reports, not updating)</td>
</tr>
<tr>
<td>8</td>
<td>Static structure; variable contents</td>
<td>Flexible structure</td>
</tr>
<tr>
<td>9</td>
<td>Small amount of data used in a process</td>
<td>Large amount of data used in a process</td>
</tr>
<tr>
<td>10</td>
<td>High availability</td>
<td>Relaxed availability</td>
</tr>
<tr>
<td>11</td>
<td>Managed in its entirety</td>
<td>Managed by subsets</td>
</tr>
</tbody>
</table>

1.5 DIFFERENCES BETWEEN ENTERPRISE DATA WAREHOUSE AND A DATA MART

To understand correctly the project background and project goal for this thesis, the concept of a data mart is discussed now in more detail. A data mart is basically another form of a data warehouse; however, it is a decentralized subset of data found in the data
A more formal definition of a data mart is as follows: “A data mart is a persistent physical store of operational, aggregated, and statistically processed data that supports business people in making decisions based primarily on analyses of past activities and results.”

According to Ralph Kimball, ‘data warehouse is nothing more than a union of all the data marts’ [1]. This, in itself, tells that data marts are the basis of the entire data warehouse project, if one were to follow a “bottom-up” design methodology for the data warehouse development (see Chapter 2, Section 2.1). In addition, it can be built with low initial investment; hence, medium level companies or small organizations can also afford to build data marts. On the other hand, building an enterprise data warehouse could be suitable to support a large business or corporate group, since there is a large initial investment required for such a development. A few of the major differences between a data warehouse and a data mart are listed below:

1. Enterprise data warehouses contain larger amounts of information. The information that is held by data marts is often available in a condensed form.
2. A data mart supports decisions concerning a specific product or production line, whereas an entire data warehouse can support broad level business decisions.
3. Data warehouse and data mart could be structurally different.
4. Data warehouse is a vast concept that covers various aspects of a corporate world. In contrast, a data mart can be easily and quickly designed and implemented for the end-users’ use.
5. With a data mart, one can see a limited view of a specific data, as it is limited to some specific department. With an entire data warehouse, one can see the overall view of the corporate data and understand inter-departmental effects of decisions.

This list could go on and on. However, while passing on, one relevant point to be noted here is that the developments of data warehouses and data marts are mutually exclusive, alternative strategies, and with proper initial planning and architectural design (see Chapter 3) both can be implemented very successfully.

1.6 BUILDING A DATA MART INCREMENTALLY

The building of a data warehouse or a data mart could be often a lengthy return on investment and a high risk project. Hence, one needs to choose a more secure and step by step approach to build such projects. Additionally, proper assessment of risks is necessary in
such projects. If an “incremental approach” is adopted, some difficulties in this activity can be reduced. Key features of this approach are:

- Always ensure that an incremental growth of the data mart is justified by identifiable and critical business needs and requirements, so as to deliver business value to the end-users of the data mart, as speedily as possible.
- This can help ensure a much earlier return on investment, and build initial confidence in the system.
- Plan the software and hardware architecture at the start of the project to avoid the cost and risk of radical changes imposed by scalability limitations.
- Make sure that all business goals and business requirements are defined and agreed upon by all key people in the particular department. If possible, create a formal document capturing these requirements.

1.7 Project Background

Building a data mart is not as simple as building any database inventory or gathering system information from various resources [3,9]. It is based on correct identification of business requirements followed by a series of critical processes which include extraction and processing of data, transformation of data, and loading of data into a well designed database or data table. If in the future any particular company grows and wants to build multiple interacting data marts (i.e. an entire data warehouse), then the architecture design should be sustainable to evolve in response to such business and technology needs; the architecture needs to be more extensible and flexible.

In this project, my objective is to build a data mart architecture and design and implement a database model using Microsoft SQL Server 2005. As an aid in this modeling, to illustrate the data mart design and implementation process, sales data information of a fictitious company, CalQ Inc., simulated in the standard representative form used by real-world companies, has been used as a background.

CalQ Corporation started as a manufacturer of cellular telephones. Currently it has 1000 employees. It quickly expanded to include a broad range of telecommunication products. As the form and size of its suite of products grew, CalQ closed down distribution channels and opened its own sales outlets.

In the past year CalQ opened a new plant, sales offices, and stores in response to increasing customer demand. With its focus firmly on expansion, the corporation put little effort into measuring the effectiveness of the expansion. Subsequently, CalQ’s growth has
started to level off, and therefore, the management is refocusing on the performance of the organization. However, although cost and revenue figures are available for the company as a whole, little data is available at the sales outlet level regarding cost, revenue, and the relationship between them.

To rectify this situation, management has requested a series of reports from the Information Technology (IT) department. IT responded with a proposal to implement a data mart and then develop a data warehouse, step-by-step. After consideration of the potential costs and benefits, management agreed to build a data mart for the product sales department for a startup.

1.8 PROJECT GOAL

Currently, CalQ Inc. has a standard E-R data base system to keep track of sales information and a basic manual query based report processing system. The main objective of this project is to build a data mart architecture and a data model to facilitate the analysis of the sales report (of the products) for the sales department of this cellular telephone manufacturing company; in other words, “build a sales dash board.” Simultaneously, the objective is to initiate and build a more flexible and efficient Sales Information Repository (SIR). This is a short term goal of this project. This data repository would allow business people to analyze the products being sold and the corresponding stores, days of sale, and promotional conditions and quantities, and generate other weekly data analysis reports.

If this short term goal is successfully achieved, the company will be planning to build multiple data marts, and, in the future, an entire data warehouse. This is one of the long term goals of this project. Hence, discussion of a generalized architecture strategy for building a data warehouse is also covered in Chapter 3.

1.8.1 Sales Information Repository (SIR) Goals

From the data mart’s architectural point of view, the new SIR should serve business reporting more effectively, centralize all the data related to product sales, and improve performance for data queries and user access.

From the point of view of data access requirements, SIR should keep proper track of the historical information to support sales analysis and forecasting.
From a functional point of view, the new SIR should improve data accuracy and consistency with current sources, reduce data redundancy, establish appropriate security procedures with a strong consideration towards ease of administration, and improve data access to enable strategic decisions.

### 1.8.2 Project Scope

The project shall be limited to direct costs, prices and revenues associated with products which are sold in USA. At a future time, rules for allocation of manufacturing and overhead costs may be created. So, the data warehouse should be sufficiently flexible to accommodate future changes. If needed (as the project evolves), IT will create a team consisting of one data analyst, one process analyst and one sales regional manager for the project.

This project is supposed to produce a detailed (data mart) modeling schema for the sales department of CalQ Inc., and discuss a generalized architecture for building the entire warehouse, which will allow integration of data marts as needed.

### 1.8.3 Anatomy of a Sale

There are two types of sales outlets: corporate sales offices and retail stores. A corporate sales office sells only to corporate customers. Corporate customers are charged the suggested wholesale price for a model, unless a discount is negotiated. One of CalQ’s twenty sales representatives is assigned to each corporate customer.

A customer can place orders through a representative or by phoning an order desk at a corporate sales office. Orders placed through a corporate sales office are shipped directly from the plant to the customer. A customer can have many shipping locations. It is possible for a customer to place orders from multiple sales offices if the customer’s policy is to let each location do its own ordering.

The corporate sales office places the order with the plant closest to the customer’s shipping location. If a customer places an order for multiple locations, the corporate sales office splits it into an individual order for each location. A corporate sales office, on average, creates 500 orders per day, five days per week. Each order consists of an average of eight product models.
A retail store sells over the counter. The suggested retail price is charged, unless a discount is negotiated. Although each product sale is recorded on an order, the company does not keep records of customer information for retail sales. A store can only order from one manufacturing plant. The store manager is responsible for deciding which products will be stocked and sold from his or her store. A retail store, on average, creates 100 orders per day, seven days per week. Each order consists of an average of two product models.

1.9 Project Approach

The project will be developed using the ‘bottom-up’ design approach suggested by Ralph Kimball (see Chapter 2). A full scale commercial development may have the following steps which are based on the star schema methodology [1,10] and are given here for illustrative purposes. The scope of the project is to discuss items 2, 3 and 5 below in detail, and discuss items 1, 4, 6, 7 and 8 in brief.

1. Create program management plan
   a. Define business strategy
   b. Define technical strategy
   c. Define procedures
   d. Develop technical and data architecture

2. Define data mart requirements
   a. Understand business needs and goals
   b. Collect specific requirements from client
   c. Document the requirements
   d. Interview clients

3. Create logical and physical view of dimensional model
   a. Provide source models and metadata for department conformed dimensions and conformed facts
   b. Work with the business analyst and subject matter experts to develop the logical view of the dimensional model, based on the approved documented data requirements
   c. Define data access and security
   d. Assess project deployment risks and come up with risk treatment plan

4. Identify the source system
   a. Analyze source system
b. Develop dataflow hierarchy  
c. Plan ETL, based on data in source system  

5. Get data inside data mart  
   a. Refine data mart requirements and categorize them  
   b. Create data modeling schema  
   c. Create error handling mechanism  
   d. Create ETL process specification  
   e. Create test scripts  
   f. Implement metadata  

6. Implement in data mart  
   a. Create project plan  
   b. Create deployment schedule including tasks, timelines and resources  
   c. Configure system and software in secure area  
   d. Migrate components into production  
   e. Production schedule and process automation  

7. Test data mart  
   a. Write and execute user test scripts  
   b. Maintain log for client request updates  
   c. Build testing team  

8. Management, administration and operation  
   a. Prepare maintenance and support plan  
   b. System management  
   c. Service management  
   d. Data management  
   e. Manage changing of dimensions  

1.10 COMMON PITFALLS TO AVOID WHILE BUILDING A DATA MART  

Below is the list of some pitfalls, in descending order, which are quite lethal — one alone may be sufficient to bring down the data warehouse/mart initiative [1,5].  

1. Allocate energy to construct a normalized data structure, yet run out of budget before building a viable presentation area based on dimensional models.  

2. Pay more attention to back-room operational performance and ease of development than to front-room query performance and ease of use.
3. Make the supposedly query-able data in the presentation area overly complex. Database designers who prefer a more complex presentation should spend a year supporting business users; they would develop a much better appreciation for the need to seek simpler solutions.

4. Populate dimensional models on a standalone basis without regard to the data architecture that ties them together using shared, conformed dimensions.

5. Load only summarized data into the presentation area’s dimensional structures.

6. Presume that the business, its requirements and analytics, and the underlying data and the supporting technology are static.

7. Neglect to acknowledge that data warehouse/mart success is tied directly to user acceptance.
CHAPTER 2
DATA WAREHOUSE CONCEPTS AND STRATEGIES

This chapter discusses strategies for building data warehouses. Basically there are three kinds of strategies currently popular in the field: (a) dimensional modeling; (b) third normal form modeling; (c) a hybrid of (a) and (b) – they take advantage of the fast turn-around time of bottom-up design and the enterprise-wide data consistency of top-down design.

2.1 TWO MAIN PARADIGMS IN THE FIELD OF DATA WAREHOUSING

In the field of data warehousing, there are two basic modeling strategies, each of which has its own way of building and implementing data warehousing. The sections below discuss these strategies briefly.

2.1.1 Bill Inmon’s Paradigm: Third Normal Form Data Model

Bill Inmon supports the third normal form data model, which is also known as the snowflake schema. He sees the data warehouse as an integral part of the corporate information factory (CIF). Rather than capture hierarchies and relationships in the dimensional table, the third normal form normalized the dimension tables by the attribute level, with each smaller dimension table pointing to an appropriate aggregate fact table [6]. His approach is also well-known as the top-down design approach.

According to this approach, data warehouse is one part of the entire business intelligence system and the data warehouse serves as the “single source of truth” for all fact and dimension data within that organization [8]; an enterprise has a single data warehouse, and data marts source their information from the data warehouse (where information is stored in 3rd normal form). Figure 2.1 shows a sample model of the snowflake schema.
Moreover, Bill Inmon emphasized the importance of a cross-functional slice of data drawn from multiple sources to support a diversity of needs [2,3]. The foundation of his subject oriented design was an enterprise data model.

2.1.2 Ralph Kimball’s Paradigm: Dimensional Data Model

Ralph Kimball introduced the dimensional data model which is based on the concepts of multidimensional databases (see Chapter 4). The most popular schema for implementing the dimensional data model is known as “star schema.” It has also been called data cube, data list, and grid file.

As shown in the diagram below, there is a fact table in the middle (a fact is an event, transaction or something that happens at a single moment in time). Surrounding the fact table are dimensional tables. Each dimensional table holds all the permutations of a single hierarchy of the company. This approach is also called as bottom-up design approach where data marts are first created to provide reporting and analytical capabilities for specific business processes. These data marts can eventually be joined together to create a comprehensive data warehouse. According to this approach, a data warehouse is the conglomerate of all data marts (subsets of data warehouse) within the enterprise [1].

Ralph Kimball introduced the notation of dimensional modeling [1] (as shown in Figure 2.2), which addresses the gap between relational databases and multidimensional databases, needed for a decision support system. Table 2.1 shows a comparison of the characteristics of each model.
Table 2.1. Comparison of Characteristics between Dimensional Data Modeling and 3rd Normal Form Modeling

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Favors Kimball: Dimensional Model</th>
<th>Favors Inmon: 3rd Normal Form Model (CIF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nature of the organization’s decision support requirements</td>
<td>Tactical</td>
<td>Strategic</td>
</tr>
<tr>
<td>Data integration requirements</td>
<td>Individual business areas</td>
<td>Enterprise wide integration</td>
</tr>
<tr>
<td>Scalability</td>
<td>Need to adapt to highly volatile needs within a limited scope</td>
<td>Growing scope and changing requirements are critical</td>
</tr>
<tr>
<td>Persistency of data</td>
<td>Source systems are relatively stable</td>
<td>High rate of change from source systems</td>
</tr>
<tr>
<td>Staffing and skills requirements</td>
<td>Small teams of generalists</td>
<td>Larger team(s) of specialists</td>
</tr>
<tr>
<td>Time to delivery</td>
<td>Need for the first data warehouse</td>
<td>Organization's requirements allow for longer start-up time</td>
</tr>
<tr>
<td>Cost to deploy</td>
<td>Lower start-up costs, with each subsequent project costing about the same</td>
<td>Higher start-up costs, with lower subsequent project development costs</td>
</tr>
<tr>
<td>Structure of data</td>
<td>Business metrics, performance measures, and scorecards</td>
<td>Non-metric data that will be applied to meet multiple and varied information needs</td>
</tr>
</tbody>
</table>
2.2 Why use Star Schema to Build the SIR?

As per the discussion in Table 2.1, following the dimensional modeling way to build SIR is the best solution, given the scope of the project. Four main reasons of choosing this approach are as follows:

1. Low initial investment;
2. Simple ‘star schema’ architecture to build data mart for single business area;
3. Ability to provide a quick solution for CalQ’s sales reporting system (short term);
4. Use SIR and its process as an example putting together an effective data warehouse infrastructure for integration and management of all CalQ’s data from different departmental operational systems.

The sales data mart forms only one part of the entire data warehouse architecture (see Chapter 3). In fact, project goals and high profile responsibilities for data warehouse bring more challenges for the design of its architecture, development and production support. Hence, while creating an architectural strategy, keeping both long term and short term goals in mind is very important.

2.3 Summary

There are mainly two basic modeling strategies in the field of data warehousing: (1) Bill Inmon’s third normal form, and (2) Ralph Kimball’s star schema. There are pros and cons for both of these strategies, and which one to choose really depends upon the business requirements and project scope. The important reasons to choose Ralph Kimball’s star schema strategy are the given scope of the project and critical time deadlines to achieve short-term goals.
DATA WAREHOUSE AND DATA MART
ARCHITECTURE STRATEGY

Data warehouse and data mart architecture should be flexible enough to integrate all the data information from a given business system and should help to successfully represent the enterprise business requirements. Hence, data warehouse designers believe that data warehouse architecture plays a key role in determining the fate of performance measurement and the entire data warehouse output [1,9].

A major problem in building a data warehouse or data mart is not the technology but rather the way in which that technology is applied to a given organizational system [1,5]. Eventually, all varying technology will fade away and will be replaced with a new one. However, a well designed data warehouse should continue to survive. All of this is possible provided the architecture of the data warehouse is well-defined initially. For the designing of a successful data warehouse, the first emphasis should be given on underlying and commitment strategy, and planning, business processes and services, which develop, deploy and maintain the data warehouse technology, and then on raw technology, which embodies the concept of data warehousing.

There are different data warehouse architectural strategies, suggested by different architects to approach the same data warehouse design; but, unfortunately, no single standard suitable strategy exists that encompasses all the data warehouse architecture designs. However, a common guideline still exists and is recommended to follow in order to achieve the data warehouse design goal. In the early 1990s, Ralph Kimball introduced the concept of the bus architecture strategy which follows the incremental approach to build the entire data warehouse. In Section 3.1, I discuss this approach in some detail.

3.1 TOP LEVEL BUS ARCHITECTURE AND MATRIX SOLUTION

The word bus is an old term from the electrical power industry that is now used commonly in the computer industry. A bus is a common structure to which everything
connects and from which everything derives power. The bus in a computer (PC OR Computer Machine) is a standard interface specification that allows plugging in a disk drive, CD-ROM, or any number of other specialized cards or devices. And because of the computer’s bus standard, these peripheral devices work together and usefully coexist, even though they were manufactured at different times by different vendors [1,5]. Figure 3.1 shows the sample data warehouse bus for a typical organization.

Figure 3.1. Sharing dimension – data warehouse bus.

The data warehouse bus architecture provides a rational approach to decentralization of data warehouse planning tasks. During the limited duration of the architecture phase of this project and the limited overall time frame, it is very important to come up with designs of a master suite of standardized dimensions and facts that have uniform interpretation across the enterprise which establishes the data architecture framework [5]. (Tackling the implementation of separate data marts, in which each iteration closely adheres to the architecture, is the next step in the process.)

As the separate data marts like sales, manufacturing, order management etc. come on line, they fit together like the pieces of a puzzle [1,5]. The bus architecture helps in putting together an architectural framework that guides the overall data warehouse design, and divides the problem into bite-sized data mart chunks which can be implemented in realistic time frames. Separate data mart development should follow the architecture guidelines while
working fairly independently and asynchronously. Moreover the bus architecture is independent of technology and the database platform [1,5].

### 3.1.1 Data Warehouse Bus Matrix

The tool used to create, document, and communicate the bus architecture is the data “warehouse bus matrix,” suggested by Ralph Kimball. I have illustrated this in Table 3.1 [5].

<table>
<thead>
<tr>
<th>Common Dimensions</th>
<th>Sales</th>
<th>Retail Inventory</th>
<th>Retail Deliveries</th>
<th>Purchase Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Product</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Store Location</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Promotion</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Transaction Type</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Contract</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Price</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Shipper</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Working in a tabular fashion, the business processes of the organization lay out as matrix rows. It is important to remember that here the business processes are closely identified with sources of data, not the organization’s business departments. The matrix rows translate into data marts based on the organization’s primary activities. Data marts that are derived from a single primary source system are commonly known as first-level data marts. These data marts are recognizable complements to their operational source. For this project, I will be starting with sales data mart (as per project scope and definition). This way, I can minimize the risk of signing up for an implementation that is too ambitious and provide users with enough interesting data to keep them happy.

The columns of the matrix represent the common dimensions. The shaded cells indicate that the dimension column is related to the business process row. The resulting matrix is surprisingly dense. Looking across the rows is revealing because one can see the
dimensionality of each data mart at a glance. However, the real power of the matrix comes from looking at the columns as they depict the interaction between the data marts and common dimensions.

The matrix is a very powerful device for both planning and communication [5]. Although it is relatively straightforward to lay out the rows and columns, in the process, it defines the overall data architecture for the warehouse. Moreover the matrix table helps to prioritize which dimensions should be tackled first for conformity given their prominent roles in the project and allows us to communicate effectively within and across different data marts.

### 3.1.2 Conformed Dimensions

As CalQ’s business people are willing to agree on a common definition for product and customer and other key terms, it better to define conformed dimensions for this project. Conformed standardized dimensions serve as the cornerstone of the warehouse bus. Conformed dimensions are either identical or strict mathematical subsets of the most granular, detailed dimension. They have consistent dimension keys, consistent attribute column names, consistent attribute definitions, and consistent attribute values (which translate into consistent report labels and groupings). Dimension tables are not conformed if the attributes are labeled differently or contain different values. If a customer or product dimension is deployed in a non-conformed manner, then either the separate data marts cannot be used together, or worse - attempts to use them together will produce invalid results [1,5].

Conformed dimensions come in several different flavors. At the most basic level, conformed dimensions mean exactly the same thing with every possible fact table to which they are joined. For example, the date dimension table connected to the sales facts is identical to the date dimension table connected to the inventory facts in the inventory data mart. In fact, the conformed dimension may be the same physical table within the database. For this project, product and date dimensions are defined as conformed dimensions.

Most conformed dimensions are defined naturally at the most granular level [1,5,11]. For this project, the grain of the product dimension is the lowest level at which products are tracked in the source systems, and the grain of the date dimension is the individual day.
3.1.3 Conformed Facts

In Section 3.1.2, I discussed the central task of setting up conformed dimensions to tie data marts together. This is 90 percent of the up-front data architecture effort. The remaining effort goes to establish conformed fact definitions.

Revenue, profit, standard prices, standard costs, measures of quality, measures of customer satisfaction, and other key performance indicators (KPIs) are facts that must be conformed [5]. In general, fact table data is not duplicated explicitly in multiple data marts. However, if facts do live in more than one location, such as in first-level and consolidated marts, the underlying definitions and equations for these facts must be the same if they are to be called the same thing. If they are labeled identically, then they need to be defined in the same dimensional context and with the same units of measure from data mart to data mart.

In this project, I will be disciplining data naming practices (see Chapters 4 and 5) and making sure that I am assigning different names to different interpretations if it is impossible to conform a fact exactly.

3.2 DATA MART REQUIREMENTS AND DATA SOURCE

As already discussed, I am building a sales data mart to meet the primary goal. For this, I need to identify the requirements and functions that the data mart will be delivering. In addition to the features and functions needed, the requirements will clearly describe the operating environment in which the data mart will be delivered [12].

The amount of requirements to be gathered and sorted usually depends upon the implementation approach and the time deadlines one is adopting for developing the data mart. The primary purpose of collecting end user requirements for a sales data mart is to understand how users conduct their business, what data they currently use, and what they would like to do in the future. Then further decomposition of this information will be done into business entities and their attributes and relationships between the entities [2,12].

The requirements can be gathered through a series of interviews with the different users. Answers to the following questions from different users will generate the requirements needed for the development of the data mart.
3.2.1 Executive/Owner Requirements

Detailed answers to all of the following questions are collected and carefully recorded into a document.

- Why are we building a data warehouse / data mart? What business problem will it address? – Sales reporting and data analysis.
- How much will it cost? – $5,000.
- When will it be ready? – 4 months.
- What is the impact on people? Skills? Organization? – People will be able to analyze the historical data correctly and plan the future sales more accurately.
- What does it do to our current computer investment? – We need 1 windows based personal computer and one server machine.
- Do we have the skills to do it? – Yes.
- What are the risks? – Investment of money, time and resources.

3.2.2 End User Requirements

Detailed answers to all of the following questions are collected and carefully recorded into another document.

- How does the data warehouse's functionality fit the end user's daily workflow?
- What are your query requirements?
  - What types of ad hoc analysis do you do? – Comparing the different attributes with each other in multidimensional data model.
  - What level of detail is included? – Daily sales report with respect to different attributes like product, store location, etc.
- What are your reporting requirements?
  - What reports do you create? – Web based comparison and listing reports.
  - How often do you perform this analysis? – If required, could be performed daily.
  - Who gets the information? – Sales manager and other managerial people in the company.
  - How is it used? – To create analysis reports.
  - Where do you get the information from? – From each store where the product is delivered.

At the end of each interview, each candidate is asked to develop a wish list which would include things that they would want to be able to do if there were no financial or time or technical constraints in their way. Next, we develop a cost requirement.
3.2.3 Cost Requirements

Traditional cost assessment is not applicable in developing a data mart because it uses an iterative development process, thereby making it impossible to predict end-user's summaries and changes [12]. A percentage of at least 10% should be allocated to metadata management throughout the life-cycle of the warehouse. Table 3.2 recommends cost allocation percentages [12].

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Percentages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform (process/disk)</td>
<td>15% - 50%</td>
</tr>
<tr>
<td>DBMS</td>
<td>5% - 25%</td>
</tr>
<tr>
<td>Warehouse Management</td>
<td>5% - 10%</td>
</tr>
<tr>
<td>Metadata Management</td>
<td>5% - 10%</td>
</tr>
<tr>
<td>Log Processing Software</td>
<td>0% - 10%</td>
</tr>
<tr>
<td>Analysis Platforms</td>
<td>10% - 20%</td>
</tr>
<tr>
<td>Analysis Software</td>
<td>1% - 5%</td>
</tr>
<tr>
<td>Middleware</td>
<td>1% - 15%</td>
</tr>
<tr>
<td>Selection/Discovery</td>
<td>1% - 5%</td>
</tr>
<tr>
<td>Data Models</td>
<td>10% - 20%</td>
</tr>
<tr>
<td>CASE/dd/Repository Interface</td>
<td>0% - 1%</td>
</tr>
</tbody>
</table>

3.3 Data Mart Architecture and Architectural Components

Considering the most recent answers to questions of business requirements from the sales team and the following dimensional modeling approach, the main components of the sales data mart architecture are listed below and shown in Figure 3.2.

1. External data sources and internal data sources;
2. Data staging and data transfer area;
3. Operational source system (OSS) or operational data store (ODS);
4. Data presentation area;
5. Data access area/data access tool.
The source systems are placed outside the data warehouse because, presumably, we have little to no control over the content and the format of the data in these operational (legacy) systems [1,5]. The main priorities of the source systems are processing performance and availability. Each source system is often a natural stovepipe application, where little investment has been made to sharing common data such as product, customer, geography, or calendar with other operational systems in the organization.

For the sales data mart, the main external data source is transactional data collected at each store. This data is sometimes in the form of CSV files, excel files, or sometimes in SQL data tables.

As an internal data source, there are some historical excel files related to the sales department which tells how sales have grown or went down for the first few years. According to the business people in the sales department, manual analysis of these files helped them to do future planning. Hence, keeping this data is important, and storage into the data mart is essential. I will be reengineering source systems for generating a consistent view at the end.
3.3.2 Data Staging Area

The data staging area is everything between the source system and the operational data store, and involves operations like cleansing the data (correcting misspellings, resolving domain conflicts, dealing with missing elements, or parsing into standard formats), combining the data from multiple sources, de-duplicating the data, and assigning warehouse keys. It is the storage area for sources and a set of process commonly known as Extract-Transform-Load (ETL). Extracting implies reading and understanding the source data and copying the data needed for the data mart into the staging area for further manipulation. It is the very first step for getting data into the data warehouse environment. The second step is to transform the data, which involves applying a series of business rules or functions to the extracted data from the source to derive the data for loading into the end target. These transformations are all precursors to loading the data into the operational data store, which is the final and third step. To summarize, in this stage raw operational data is transformed into warehouse deliverable data which is fit for user query. Moreover, conformed facts and dimensions are built in this area.

3.3.3 Operational Data Store (ODS)

ODS is the main repository for a data warehouse or data mart where all clean and consistent data is stored. Data provided by ODS is also partially used by the maintenance application (see Section 3.4). ODS is frequently updated to keep track of all the recent data. One of the benefits of ODS is that the end user’s queries do not degrade the performance of the OLTP system (see Appendix A). It also helps in making tactical decisions which need an integrated set of data and has characteristics of being both integrated and subject-oriented. The Operational Data Store provides data to a data mart for querying and data processing.

3.3.4 Data Presentation Area

The data presentation area is where data is organized, stored, and made available for direct querying by users, report writers, and other analytical applications. Since the backroom staging area is off-limits, the presentation area is the data warehouse or data mart as far as the business community is concerned. This is all that the business community sees and touches via data access tools. It consists of single or multiple data marts. The SIR would contain
detailed, atomic data, adhere to the data warehouse bus architecture and pull out data from ODS.

3.3.5 Data Access Area and Data Access Tool

We use the term *tool* loosely to refer to the variety of capabilities that can be provided to business users to leverage the presentation area for analytic decision making. By definition, all data access tools query the data in the data warehouse’s presentation area, and querying, obviously, is the whole point of using the data warehouse.

A data access tool can be as simple as an ad hoc query tool or as complex as a sophisticated data mining application. For this project, I am using SQL Server Management Studio 2005 Express and the Microsoft SQL language for querying the data tables in the sales data mart.

3.4 DATA MART LIFE CYCLE

Figure 3.3 shows the general data mart life cycle which is divided into nine sub parts [10]. The diagram is self-explanatory and has two main phases: (a) design and development, where the actual sales data mart will be getting developed; and (b) enhancement, where new improvements, changed requirements and suggestions are taken care of post-verification.

3.5 DATA FLOW AND DATA ACQUISITION STRATEGY

The process of moving the operational data from different remote external sources to a data warehouse environment involves the most critical strategy in building data marts and data warehouses. This strategy is usually called a *data acquisition strategy*, which includes both data flow from source to data access tool, and the data acquisition process.

In this project data extraction from the remote external sources is done when operational system (transactions at stores) is relatively slow. Based on current business requirements, the data extraction process should be done repetitively.

3.5.1 Data Flow

Figure 3.4 (p. 27) shows the data flow sequence in the sales information repository (SIR) which is mainly based on ETL processes.
Figure 3.3. Data mart life cycle.
1. External source to staging area: Historical data will be coming from excel files in the internal data source. The data is extracted from the external data source carefully and loaded to the staging area using ETL. This will be an ongoing process as per user requirement. It is not always the case that the data moved from the external sources to the data warehouse environment must involve many changes to the source data to fit the extraction process.

2. Data staging area to ODS: This is done only for initialization of the ODS (i.e. consolidation of data). Extraction of data is done again in this phase from the staging area and integrated into ODS. Data in ODS is then updated for all subject areas.

3. ODS to sales data mart: Here, one time load from ODS to sales data mart is done, which includes data from April 2004 (when the company started), through the current date. Data mart also gets the data from the metadata store whenever required.

4. Maintenance Application to ODS: To update, insert and delete, I have added a maintenance application which is extracting data partially from ODS and feeding it back to ODS.

3.5.2 ETL Tools

There are many ETL tools currently in the market which promise to save organizations time and money. Each of these tools has its own advantages and disadvantages. Choice of the best fit tool for specific organizational needs depends upon the business requirements and organizational culture. Below I summarize the advantages and disadvantages of some of the popular tools [13].
3.5.2.1 **Informatica PowerCenter**

**Advantages:**
- Most substantial size and resources on the market of data integration tools vendors;
- Consistent track record, solid technology, straightforward learning curve, ability to address real-time data integration schemes;
- Informatica is highly specialized in ETL and data integration and focuses on those topics, not on BI as a whole;
- Focus on B2B data exchange.

**Disadvantages:**
- Several partnerships diminishing the value of technologies;
- Limited experience in the field.

3.5.2.2 **Microsoft (SQL Server Integration Services)**

**Advantages:**
- Broad documentation and support, best practices to data warehouses;
- Ease and speed of implementation;
- Standardized data integration;
- Real-time, message-based capabilities;
- Relatively low cost - excellent support and distribution model.

**Disadvantages:**
- Problems in non-Windows environments. Takes over all Microsoft Windows limitations;
- Unclear vision and strategy.

3.5.2.3 **Oracle (OWB and ODI)**

**Advantages:**
- Based on Oracle Warehouse Builder and Oracle Data Integrator – two very powerful tools;
- Tight connection to all Oracle data warehousing applications;
- Tendency to integrate all tools into one application and one environment.

**Disadvantages:**
- Focus on ETL solutions, rather than an open context of data management;
• Tools are used mostly for batch-oriented work and transformation rather than real-time processes or federation data delivery;
• Long-awaited bond between OWB and ODI brought only promises - customers confused in the functionality area; the future is uncertain.

3.5.2.4 SAP BUSINESS OBJECTS (DATA INTEGRATOR/DATA SERVICES)
Advantages:
• Integration with SAP;
• SAP Business Objects created a firm company determined to stir the market;
• Good data modeling and data-management support;
• SAP Business Objects provides tools for data mining and quality profiling due to many acquisitions of other companies;
• Quick learning curve and ease of use.
Disadvantages:
• SAP Business Objects is seen as two different companies;
• Uncertain future. Controversy over deciding which method of delivering data integration to use (SAP BW or BODI);
• Business Objects Data Integrator (Data Services) may not be seen as a stand-alone capable application to some organizations.

3.5.3 Data Acquisition Process
Data acquisition is one of the compulsory processes when building a data mart. First, every table in the data mart has to go through the preliminary process of data loading which loads data from files and places the data in tables within data mart. For this project, the initial loading process feeds old data from the existing excel files from the internal data source to the staging area, ODS and then the data mart. Next is the affixing (appending) load process, which is specifically for larger tables that accumulate a lot of historical records over time, and which do not allow updates to existing data [5,11]. Finally, if needed, is the delete load process, which deletes the old record in the data mart. Sometimes, a partitioning technique is used to do this step if data to be deleted is in a really large quantity (see Chapter 5).

As per the customer demand or business requirement, customers many times require to see very current data information. To facilitate this, one needs to refresh the data mart. This process is referred to as data acquisition refreshment.
3.6 Metadata Management

Typically, metadata is “data about data” or “information about information.” It helps to better understand and interpret in a descriptive manner, the actual data. Metadata is akin to an encyclopedia for the data warehouse.

The primary objective for the metadata management process is to provide a directory of technical and business views of the data mart metadata. Metadata comes in a variety of shapes and forms to support the disparate needs of the data warehouse’s technical, administrative, and business user groups and can be mainly categorized as either technical metadata or business metadata [14].

Technical metadata consists of metadata created during the creation of the data mart, as well as metadata to support the management of the data mart. This includes data acquisition rules, the transformation of source data into the format required by the target data mart, and schedules for backing up and refreshing data. Business metadata allows end users to understand what information is available in the data mart and how it can be accessed. For example, it keeps information about which user groups need what kind of specific information from the data mart.

In this project, I have provided external and internal source system metadata including source schemas and copybooks that facilitate the extraction process. This metadata is stored in SQL server data modeling schema. Once data is in the staging area, I encounter staging metadata to guide the transformation and loading processes, including staging file and target table layouts, transformation and cleansing rules, conformed dimension and fact definitions, aggregation definitions, and ETL transmission schedules and run-log results.

At the end, having good metadata significantly improves the value of the data mart and the entire data warehouse. The ultimate goal is to corral, catalog, integrate, and then leverage these disparate varieties of metadata, much like the resources of a library. For managing this metadata, the only way in this project is to use PL/SQL language and keep track of the updated data about data.

3.7 Summary

Data warehouse and data mart architecture should be flexible enough to integrate all the data information from a given business system and help to successfully represent the
enterprise business requirements. By defining a standard bus interface for the data warehouse environment, separate data marts can be implemented by different groups at different times. The separate data marts can be plugged together and usefully coexist if they adhere to the standard.

Creating the data warehouse bus matrix is one of the most important up-front deliverables of a data warehouse or data mart implementation. Here, I have used Ralph Kimball’s bus architecture and matrix model to develop the sales data mart architecture design. In addition, I have identified the main data sources and also set up a data acquisition strategy for the SIR. To meet short term and long term goals, architectural details and conformed dimensions and facts are discussed in detail.

Finally, four keys to the data mart project planning and management include: (1) having a solid business sponsor; (2) balancing high value and do-ability to define the scope; (3) developing a detailed project plan; and (4) performing acceptance testing.
CHAPTER 4

DIMENSIONAL DATA MODELING

Data modeling can be formally defined as the process of generating an internally consistent conceptual schema of the enterprise or problem environment, which is a valid representation of both the set of real world and things deemed relevant to the enterprise, and of the business rules which control the enterprise’s interaction with these things [11,15]. It is, basically, defining an abstract, summary representation of the real world that will form the basis of the physical database design, and acts as a bridge between the real world as seen by the enterprise and the information processing system that will support the enterprise in doing its work.

Data modeling is an essential aspect of most information technology projects, since it provides a means of determining just what data is required for a successful project. Mainly, there are three levels of data models required to build a data warehouse or data mart:

1. A high level logical model, consisting of corporate subject area diagram;
2. A midlevel logical model of the business subject area applicable to a particular phase of the data mart construction;
3. The physical model, which reflects the changes necessary to reach performance objectives.

The ultimate purpose of data modeling is to assure that we are working with a complete and accurate picture of the data required by the enterprise to perform the specified set of activities which are based on high level business requirements.

4.1 ADVANTAGES OF USING DIMENSIONAL DATA MODELING

In Chapter 2, I discussed basics about the dimensional data model. This chapter discusses all the details of this model in terms of building the sales data mart and SIR. The dimensional data model provides a method for making databases simple and understandable and defines how a user will access information; it is similar to how the user thinks of information. A common way of representing this model is a “cube” of three or four
dimensions where users can access a slice of the database along any of the dimensions (see Figure 4.1).

![Figure 4.1. Multidimensional cube structure.](image)

The visualization of how data is organized in a multidimensional way is given by a cube where each dimension is associated with an identifying attribute. For instance, suppose that the factual data of a business is stored in a sales table. The analysis of sales to measure the business performance with respect to sales could be in terms of time periods, product items, store locations, sales persons, organizational setups, etc. The number of dimensions is mostly directly proportional to the level of details retrieved from the model. For this project, I have three main dimensions exhibited in Figure 4.1, which will also serve as significant criteria to achieve the long term goals for this project.

This modeling helps to identify the *dimensions* (dimensional tables) for the given problem, planning queries and to separate *facts* (fact table) to answer these queries and tells how data is going to be “sliced-and-diced” down at the time of report generation. For planning these queries correctly, declaring the *grain* for the data mart is very important. Declaring the grain means specifying exactly what an individual fact table row represents. The grain conveys the level of detail associated with the fact table measurements and it answers the question: “What level of data detail should be made available in the dimensional model?” [1, 5].
For this project, the most granular data is an individual line item on a Sales Force Fact table which keeps track of transactions on each line. Sales data is rolled up by product in a store on each day.

The main advantage of using the dimensional data model is for calculating summarized data. For example, for this project, sales data could be collected on a daily basis and then be aggregated to the week level, the week data could be aggregated to the month level, and so on. The data can then be referred to as aggregate data. Aggregation is synonymous with summarization, and aggregate data is synonymous with summary data. The performance of dimensional data modeling can be significantly increased when materialized views are used [5,11,15]. A materialized view is a pre-computed table comprising of aggregated or joined data from facts and, possibly, dimensional tables.

If we try to analyze the traditional relational data modeling and compare it with the dimensional data model, the main distinction is observed in multidimensional databases. Some more differences between relational data modeling and dimensional data modeling are highlighted in Table 4.1.

**Table 4.1. Comparison of Relational and Dimensional Data Modeling**

<table>
<thead>
<tr>
<th>Relational Data Modeling</th>
<th>Dimensional Data Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Data is stored in RDBMS.</td>
<td>1. Data is stored in RDBMS or multidimensional databases.</td>
</tr>
<tr>
<td>2. Tables are units of storage.</td>
<td>2. Many times, cubes are units of storage.</td>
</tr>
<tr>
<td>3. Data is normalized and used for OLTP.</td>
<td>3. Data is denormalized and used in data warehouse and data mart. Optimized for OLAP.</td>
</tr>
<tr>
<td>4. Several tables and chains of relationships among them.</td>
<td>4. Few dimensional tables are connected to fact table or tables.</td>
</tr>
<tr>
<td>5. Volatile (several updates) and time variant.</td>
<td>5. Non volatile and time invariant.</td>
</tr>
<tr>
<td>6. Detailed level of transactional data.</td>
<td>6. Summary of bulky transactional data (aggregates and measures) used in business decisions.</td>
</tr>
</tbody>
</table>
4.2 STAR SCHEMA

As shown in Figure 2.2 (Chapter 2), the star schema arranges all the entities into a star-like display. This technique restricts the complexity of the table joins. It is one of the simplest styles of data warehouse schema. The start schema is composed of two types of tables:

1. Fact table: A primary table wherein the numerical performance measurements of the business are stored. Fact tables express the many-to-many relationships between dimensions in dimensional models.

2. Dimensional Table: Dimension tables are integral companions to a fact table. The dimension tables contain the textual descriptors of the business.

Here, for designing the sales data mart for CalQ Inc., I am using star schema exclusively to restrict the complexity of each query and improve the end query performance for reporting. I have designed a single dimensional SALES_FACT table and multiple dimension tables like product, store_location, price etc. and there is a many-to-many relationship between a fact table and dimension tables (see Appendix B).

4.3 SALES DATA MART: DATA MODEL DESIGN OVERVIEW

Data modeling and designing is one of the most creative and taxing processes in building the data mart / data warehouse, since designing the data mart is an iterative process. The best way to make this process easy is to follow the guidelines and understand the business data. Building the data model for the sales data mart ensures that (1) the scope is complete, (2) the pieces of the complex data warehouse project interlock, and (3) redundancy is recognized and controlled.

4.3.1 Effect of Normalization and Denormalization on Query Performance

Both normalization and denormalization are basic concepts in the world of data bases. Key driving factors in the choice between the above methods are business and data requirements. Both of these methods have pros and cons.

Normalization is often referred as “normalization by decomposition.” Normalization theory and methods are based on the observation that a certain set of relations have better properties in an inserting, updating and deleting environment than do other sets of relations.
containing the same data [15]. It places all data related to one subject area into one non-redundant table and restricts changes to a single row in a single table. This is achieved when attributes depend upon the key.

The advantage of doing this is to decrease the efforts of data maintenance loads and save storage space, whereas a major disadvantage is that it decreases query performance as more joins are needed to execute the given query. It can be applied, to some degree, to a data model at any stage of the data mart and RDBMS development. On the other hand, denormalization is used more often for improving query performance. It is appropriate for OLTP but requires an advanced knowledge of how the data will be used. It emphasizes performance over redundancy in space allocation. In this project, the star schema model which I am using to build SIR, is heavily based on denormalization theory to optimize the query processing speed at the cost of storage space (because here, the important business requirements are related to faster query processing for quicker report generation). Besides, the cost of data storage space has reduced over the last decade. This allows me to trade-off data storage space for superior query performance.

4.3.2 Naming Conventions Overview

In modeling, a great deal of time and effort is spent in working out naming conventions for entities, attributes and relationships. It is also important to recognize physical limitations on names imposed by target implementation environments, at least during the physical design stages. These conventions help business users and developers to better understand and use business data from an enterprise perspective without much confusion.

Standardizing data names is a well proven practice which is very useful and plays key role at the moment of data integration across an enterprise. Here, I am using standard naming convention for table and attribute naming defined by Microsoft SQL server 2005. These practices will assist in: (a) transferring data structures from files to database, (b) reducing conflicting data structures and definitions, (c) integrating the business data easily, and (d) improving system and data quality.

4.4 Logical View of a Data Model

As understanding of data increases, data naturally congregates around major categories relevant to the business called subject areas or high level entities which are also
known as primary entities. For example, this project has seven subject areas shown in bold, big, rectangular boxes in Figure 4.2.

For building a logical model of the sales data mart here, I have combined mid-level and high-level logical modeling stages where entities are groups of the logical data model. At this stage, the model has ‘common business topics’, such as methods of sale, and places where the product was sold, about which the company records information.

4.4.1 Entity Definition

Modeling starts by organizing the aspects of the real world into logical classes. For instance, “Target,” “Wal-Mart,” and “Sprint Shoppe,” can be put into many legitimate classes, but for the purpose of an addressing database, they each best fit into a class of “Store Names.” Each class of things or a relationship is determined to be a part of the sales data mart system. Hence, the creation of a corresponding representation within the actual database makes logical sense. This representation is the entity.
For this project, each entity is defined independently. For example, “PRODUCT = contains all the information related to products that are manufactured by CalQ and that are currently in the market.”

### 4.4.2 Entity Naming

The entity name must be unique for the given data model and composed from business-oriented terms. Length of the entity and attribute names should be limited to 30 to 35 characters with singular nouns. All the letters in the name should be in capital letters. For example, Operational Data Store table must have a ODS prefix, Data Mart Fact table must have a DMF prefix, Data Mart Dimension table must have a DMD prefix.

### 4.4.3 Attribute Naming and Organization Format

Attribute names should generally avoid numeric characters and should be always in lower case. They should be named in singular form and should be always in present tense.

### 4.4.4 Attribute Definition

The fact is a true statement of a property [15]. The main concern in designing the database or data model is to create a place to store these facts where each fact is identified as relevant and is represented by a corresponding ‘attribute’ in the data model. When we get to the level of physical design, an attribute becomes a field in the database table. “Attribute” in simple words means a property of an entity [10,15]. I use a consistent format for the definition of all attributes, which is non-recursive and business oriented.

### 4.4.5 Data Naming Abbreviations

To shorten names to fit within the length of a restricted name, abbreviations or acronyms are used. CalQ Inc. has a set of pre-defined standard abbreviations and acronyms. Hence, I am using the same list while working on this entire project development. This list is maintained in a shareable electronic format. Hence, if needed for this project, I can just update or add to this electronic list so that all other business people in the company should also share the same.
4.5 PHYSICAL MODEL

The physical model will differ from the logical model in terms of the details specified for the physical database, including physical column names (lengthy names), data types, key declarations (if appropriate), and the permissibility of nulls.

This step finally deals with actual tables and column details which are based on entity and attribute names in the logical model. During this phase, I have translated the expected schema into an actual database structure by mapping: (a) entities to tables, (b) relationships to key constraints, and (c) attributes to columns.

4.5.1 Table Definition Guidelines and Table Naming

Table definitions and naming constraints are the same as the entity definitions, and entity naming, and should follow all the specifications.

4.5.2 Column Naming

These names should be unique within all catalogs where it is documented and always be in small case letters. They should be descriptive enough so that one knows clearly what data is stored in that column, except that the words that exist on the list of abbreviations should always be abbreviated. Mainly, alpha-numeric characters should be used for naming columns and those should be singular.

Slandered delimiters between words and data names should be used, like the underscore character ( _) in database and hyphen character in (-) in flat records.

4.5.3 Primary Key Constraint Naming

All the foreign key names should be descriptive and, if needed, should be renamed before actual database implementation. Columns which are assigned as the PRIMARY KEY should not accept null values at all.

4.5.4 Trigger Naming

Trigger naming should follow the sequence of the following format:

Trg+(action)+(column_name).
4.5.5 Aggregation Strategy

As said by experts, every data warehouse should contain pre-calculated and pre-stored aggregation tables [5,11,15]. Given stringent rules about usually avoiding mixed fact table granularity, each distinct fact table aggregation should occupy its own physical fact table. When I aggregate facts for SIR, I am either eliminating the dimensionality or associating the facts with a ‘rolled-up’ dimension.

To administer aggregation of fact tables, I have considered two primary factors:

1. Think about the business users’ access patterns. In other words, what data are they frequently summarizing on-the-fly? The answer to this question can be derived from insights obtained from the analysis of business requirements, as well as from the inputs gained by monitoring actual usage patterns.

2. Assess the statistical distribution of the data. For example, how many unique instances do I have at each level of the hierarchy, and what is the compression from one level to the next? If 7 days roll up into a week, I would be summarizing only 7 base rows (on an average) here, to calculate the weekly sales price aggregate. In such a case, it is not worth the effort to physically pre-store the aggregate.

On the other hand, if I can avoid touching 100 base rows by accessing the aggregate instead, it makes much more sense. The aggregation game here boils down to reducing the input-output. In general, the disk space required by aggregate tables should be approximately two times the space consumed by the base-level data [1,5].

The availability of an aggregate navigator is another consideration in the overall aggregation strategy of the data warehouse. Without an aggregate navigator, the number of aggregate schemas for analytical users to manually choose from is very limited — probably no more than two aggregates per base fact table. Aggregate navigator functionality sits between the requesting client and relational database management system [5]. The navigator intercepts the client’s SQL request and, wherever possible, modifies it so that it accesses the most appropriate performance enhancing aggregates. The aggregate navigator makes productive use of the aggregate tables while buffering the client applications. Clients do not need to specifically write their query to access a specific base fact table. This is not the case with an aggregated fact table, which requires that queries be rewritten when aggregates are added or dropped. The navigator handles changes to the aggregate portfolio behind the scenes so that the client can remain oblivious, as it should be. I will not be adding an aggregation navigator currently as that falls into the long term goals of this project.
4.5.6 Indexing Strategy

Dimension tables will have a unique index on the single-column primary key. Meanwhile, fact tables are the behemoths within the data warehouse. So, I needed to index them more carefully. The primary key of the fact table is almost always a subset of the foreign keys. I am typically placing a single, concatenated index on the primary dimension of the fact table. Since many dimensional queries are constructed on the date dimension, the date foreign key should be the leading index term. In addition, having the date key in the first position speeds the data loading process where incremental data is clumped by date.

4.5.7 Strategy for Handling Slowly Changing Dimensions

The "slowly changing dimension" problem is a common problem in a typical data warehouse or data mart. In particular, dimensions have been assumed to be independent of time. Unfortunately, this is not the case in the real world. While dimension table attributes are relatively static, they are not fixed forever. Dimension attributes change, albeit rather slowly, over time [1,5]. In a nutshell, this applies to cases where the attribute for a record varies over time.

There are multiple methods to handle the slowly changing dimensions. Choice of the technique depends on the project’s business requirements. Here, a surrogate key (see Chapter 5) is a non-intelligent “integer” key for each record in the table. There are three basic techniques to solve these problems. These are discussed below, along with a hybrid technique.

4.5.7.1 Type One: Overwrite Old Data with New Data

Using this method, one cannot store the history. For example, say each customer can have one sales representative (salesrep) at any given point in time. When the salesrep of CalQ Inc. changes from Sandy to Laura, the fact that Sandy was a salesrep of CalQ Inc. will not be kept stored anywhere. Any report by the salesrep will assume that Laura is the salesrep of CalQ Inc. all the time and count all the sales previously done by Sandy as those by Laura.
The situation above does not make sound business sense. However, if you only need to report the sales of the current period, and salesrep does not change during this period, one can use the method above. Many OLTP tables do not need to track the history of changes. In such cases, this method may be used by the source application.

### 4.5.7.2 Type Two: Add a New Record at the Time of the Change

Using this method, all prior history about a record can be saved. There are three alternate methods to achieve this:

1. **Method (A) – No surrogate key – Use timestamp**
   
   When a change happens, a new record is added into the table. All the attributes are copied from the previous record except the changed values. The natural key is copied as well, so the timestamp is used to differentiate the records. When a fact table is joined with the dimension, if you are interested in the historical data, the timestamp will be used as part of the joined condition. To ease the joining, the record typically uses two date columns – the effective start date and the effective end date.

2. **Method (B) – No surrogate key – Use version number**
   
   Instead of using the date column, a version number is used to differentiate the different versions of the records. This technique requires the fact table to store both natural key and the version number to retrieve a given version of the dimension date.

3. **Method (C) – Use a surrogate key**
   
   When an attribute is changed, a sequence generated key is used. The fact table will also use this key column as the foreign key.

### 4.5.7.3 Type Three: Track Changes Using a Separate Column

Using this method, one can use a separate column of a dimension table to store the values of previous years, in addition to the current year data. This method does not track all the history, but just one prior version. If the data is changed, the old value needs to be moved from the current value column to the prior column and the new value overwrites the current column. This method is used when the change is not random but at a predefined interval such as an annual change.

### 4.5.7.4 Type Four: Hybrid Techniques

These techniques are a combination of one or more basic slowly changing dimension techniques. According to experts in the field one should not pursue these techniques unless
the business agrees that they are needed to address their requirements [5,11]. I use a type-two method, which uses the time stamp for handling slowly changing dimensions in SIR.

4.6 SUMMARY

The ultimate value of data modeling is the assurance that one is working with a complete and accurate picture of the data required by the enterprise to perform the specified set of activities. The key to a successful design of the data model lies in understanding the data correctly.

For SIR model design, I have followed the dimensional modeling guideline including star schema to effectively organize and translate the business data in order to improve query performance for reporting purposes. I have successfully defined dimension and fact tables and relationships between them, where each row in a fact table corresponds to a measurement. In addition, I have discussed the strategy for handling slowly changing dimensions. Moreover, naming conventions, attributes and entity definitions are provided at the physical data model levels which are very important with respect to enterprise level data integration.
A database is an organized collection of data for one or more multiple uses. It is the environment for managing data and query functionality. Here, actual implementation and verification of the physical data model is done to decide whether the data model design can provide a solution to business requirements and needs. The database design process helps in creating the correct data model for a given database and reflects the information system, of which, the database is a part.

For this project, I used Microsoft SQL Server 2005 (with the Express management studio), the most popular and powerful database software, to create the Sales Data Mart.

5.1 SQL SERVER DATABASE CONFIGURATIONS

SQL Server can obtain a very high level of performance with relatively little configuration tuning. One can obtain high levels of performance by using good application and database design, and not by extensive configuration tuning.

The most important things, after installing SQL server 2005, are as follows: (a) configure to manage the services associated with SQL Server, (b) configure the network protocols used by SQL Server, and (c) configure to manage the network connectivity configuration from SQL Server client computers. This is achieved through the SQL Server Configuration Manager. I have configured services like Management Studio Express using this manager tool to start, stop, pause, or resume one or more services.

The SQL Server Configuration Manager (see Figure 5.1) can also be used to configure server and client network protocols, and connectivity options, by creating or removing an alias, changing the order in which the protocols are used, or by viewing the properties of a server alias, including:
Server Alias: used for the computer to which the client is connecting;
- Protocol: used for the configuration entry;
- Connection Parameters: the parameters associated with the connection address for the network protocol configuration.

SQL Server 2005 provides robust authentication features that provide better support at the security outskirts of the server for letting the good guys in and keeping the bad guys out. SQL Server Authentication provides authentication for non-Windows-based clients or for applications using a simple connection string containing user identifications (user IDs) and passwords.

For connecting to the database, using Management Studio Express 2005, I am using automated Windows authentication as the database administrator ("admin") currently. As soon the sales data mart will be in production, I will be assigning user names and passwords and using SQL server authentication for operating on the database. The password policy is always checked by default, but I will be suspending enforcement for individual logins with either the CREATE LOGIN or ALTER LOGIN statements by using the following code:

```sql
CREATE LOGIN Kathy WITH PASSWORD = 'V1v3c9Es8',
CHECK_EXPIRATION = OFF, CHECK_POLICY = OFF
ALTER LOGIN Kathy WITH PASSWORD = '3x1TqYIAz' UNLOCK
```
5.2 DATABASE BACKUP

Management studio provides multiple options and tools to operate on data in the database, one of which is job scheduling. I created a stored procedure (usp_AddScheduledJob) to take the backup of the sales database and I scheduled this stored procedure to run every other day at 11:00 pm using the job scheduler. The procedure code is shown below:

```sql
GO
DECLARE @ScheduledSql varchar(800), @IsRepeatable BIT
GO
USE [master]
BACKUP DATABASE [Sales]
TO DISK = N'c:\Sales_DM.bak'
WITH NOFORMAT, INIT, NAME=N'Sales_DM-Full Database Backup', SKIP, NOREWIND, NOUNLOAD, STATS=10
GO
SELECT @ScheduledSql = N'DECLARE @backupTime DATETIME,
@backupFile NVARCHAR(512);
SELECT @backupTime = GETDATE(),
@backupFile = ''C:\Sales_DM.bak'' +
replace(replace(CONVERT(NVARCHAR(25),@backupTime, 120),
'' '' , ''_''), '':'', ''_'') + N''.bak'';
BACKUP DATABASE Sales TO DISK = @backupFile;'
,@IsRepeatable = 0
EXEC usp_AddScheduledJob @ScheduledSql, @IsRepeatable
GO
```

5.3 ROLE PRIVILEGES

Roles are basically a named collection of related privileges that are granted to users for operating or modifying the database. SQL server Management Studio offers user friendly GUI interface for managing these privileges. There are mainly two types of roles one can assign using SQL server Management studio (shown in Figure 5.2). These roles can be added be added and removed very easily (see Appendix B).

1. Fixed database roles: these are defined at the database level and exist in each database and only members of the db_owner database role can add members to the db_owner fixed database role.

2. Application roles: enables an application to run with its own, user-like permissions. Application roles contain no members and are inactive by default and work with both authentication modes (Windows and SQL server). In addition, they cannot access server-level metadata because they are not associated with a server-level principal.
My role in the sales data mart is that of a DBA (data base administrator and db_owner) and system administrator (sa), and I have all rights to grant privileges (like update or delete tables in the database or use stored procedures) to each role and then grant the role to each member or user. Specific permissions are granted to each of these fixed database roles initially and I, as the DBA, can deny or revoke these permissions later if necessary.

Fixed data base roles and their respective database and server level permissions are listed in Table 5.1 with references to Microsoft MSDN library.

### 5.4 User Privileges

I have defined two groups of users, which are classified as datamart_guest (guest) and datamart_owner (gauree) as shown in Table 5.2. To create these users, I have again used SQL server management studio GUI interface and assigned the specific privileges to both to access SIR.
### Table 5.1. Database Roles and Their Respective Permissions

<table>
<thead>
<tr>
<th>Fixed database role</th>
<th>Database-level permission</th>
<th>Server-level permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>db_accessadmin</td>
<td>Granted: ALTER ANY USER, CREATE SCHEMA</td>
<td>Granted: VIEW ANY DATABASE</td>
</tr>
<tr>
<td>db_backupoperator</td>
<td>Granted: BACKUP DATABASE, BACKUP LOG, CHECKPOINT</td>
<td>Granted: VIEW ANY DATABASE</td>
</tr>
<tr>
<td>db_datareader</td>
<td>Granted: SELECT</td>
<td>Granted: VIEW ANY DATABASE</td>
</tr>
<tr>
<td>db_datawriter</td>
<td>Granted: DELETE, INSERT, UPDATE</td>
<td>Granted: VIEW ANY DATABASE</td>
</tr>
<tr>
<td>db_ddladmin</td>
<td>Granted: ALTER ANY ASSEMBLY, ALTER ANY ASYMMETRIC KEY, ALTER ANY CERTIFICATE, ALTER ANY CONTRACT, ALTER ANY DATABASE DDL TRIGGER, ALTER ANY DATABASE EVENT, NOTIFICATION, ALTER ANY DATASPACE, ALTER ANY FULLTEXT CATALOG, ALTER ANY MESSAGE TYPE, ALTER ANY REMOTE SERVICE BINDING, ALTER ANY ROUTE, ALTER ANY SCHEMA, ALTER ANY SERVICE, ALTER ANY SYMMETRIC KEY, CHECKPOINT, CREATE AGGREGATE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE QUEUE, CREATE RULE, CREATE SYNONYM, CREATE TABLE, CREATE TYPE, CREATE VIEW, CREATE XML SCHEMA COLLECTION, REFERENCES</td>
<td>Granted: VIEW ANY DATABASE</td>
</tr>
<tr>
<td>db_denydatareader</td>
<td>Denied: SELECT</td>
<td>Granted: VIEW ANY DATABASE</td>
</tr>
<tr>
<td>db_denydatawriter</td>
<td>Denied: DELETE, INSERT, UPDATE</td>
<td>Granted: VIEW ANY DATABASE</td>
</tr>
<tr>
<td>db_owner</td>
<td>Granted with GRANT option: CONTROL</td>
<td>Granted: VIEW ANY DATABASE</td>
</tr>
<tr>
<td>db_securityadmin</td>
<td>Granted: ALTER ANY APPLICATION ROLE, ALTER ANY ROLE, CREATE SCHEMA, VIEW DEFINITION</td>
<td>Granted: VIEW ANY DATABASE</td>
</tr>
</tbody>
</table>

### Table 5.2. Types of Users and Schema Owned by Them

<table>
<thead>
<tr>
<th>User Name</th>
<th>Owned Schemas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guest (datamart_guest)</td>
<td>db_datareader, guest</td>
</tr>
<tr>
<td>Gauree (datamart_owner)</td>
<td>db_datareader, db_datawriter, db_ddladmin, db_owner</td>
</tr>
</tbody>
</table>
End users are typically allowed to read and access the data from dimension tables rather than from fact tables. This strategy creates a great elasticity for the system administrator and the DBA, to grant different object privileges to the end user based on specific requirements, and alternative ways to enforce the data securities for sales information on SIR. For example, to create an individual user guest, I have used the following query:

```sql
CREATE LOGIN gauree
    WITH PASSWORD = 'Mcxo7788';
USE sales;
CREATE USER gauree FOR LOGIN gauree;
GO
GRANT CONNECT TO db_owner;
GO
```

## 5.5 Tables and Surrogate Keys, Indices

In SQL server, a database table holds all the user-ascrivable data. Hence, it is called a basic unit of the data storage [7,11]. Each table has a specific name and set of columns containing the data in each cell. While assigning names to columns, it is very important to define a data type for each column.

I have created eight tables in the sales database. Seven of eight are dimension tables, and one is a fact table. Syntax for each table creation is described in Appendix B. The details about the keys for each data table are shown in Table 5.3.

### Table 5.3. Data Tables and Their Respective Natural and Surrogate Keys

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Type</th>
<th>Natural Key or Business Key</th>
<th>Surrogate Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMD_CALENDAR</td>
<td>Dimension Table</td>
<td>day_number_in_month</td>
<td>Date_ID</td>
</tr>
<tr>
<td>DMD_LOCATION</td>
<td>Dimension Table</td>
<td>Store_code</td>
<td>Location_ID</td>
</tr>
<tr>
<td>DMD_PRICE_INFO</td>
<td>Dimension Table</td>
<td>Product_price_per_unit</td>
<td>Price_ID</td>
</tr>
<tr>
<td>DMD_PRODUCT_INFO</td>
<td>Dimension Table</td>
<td>Product_SKU</td>
<td>Product_ID</td>
</tr>
<tr>
<td>DMD_PROMOTION_INFO</td>
<td>Dimension Table</td>
<td>Promo_Code</td>
<td>Promo_ID</td>
</tr>
<tr>
<td>DMD_SELLING_MEDIUM</td>
<td>Dimension Table</td>
<td>Medium_code</td>
<td>Medium_ID</td>
</tr>
<tr>
<td>DMD_SERV_PROVIDER_INFO</td>
<td>Dimension Table</td>
<td>Prov_code</td>
<td>Provider_ID</td>
</tr>
<tr>
<td>DMF_SALES_FACT</td>
<td>Fact Table</td>
<td>Transaction_num, Product_SKU</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Relational databases like SQL Server use indices to find data quickly when a query is processed. It is always suitable to create indices on columns of large tables which are getting searched many times, because it can reduce the number of rows scanned by the given query. The size of the index structure should be manageable so that the benefits can be accrued by traversing such a structure.

Here, I have created an index on DMD_CALENDER, DMD_PRODUCT_INFO and DMF_SALES_FACT tables. The syntax for creating the index on DMD_PRODUCT_INFO is: CREATE INDEX [INX_Model_num] ON DMD_PRODUCT_INFO (Model_num).

For long term goals (when there will be a large number of rows in single table), I will be using the Projection Index strategy [7] to create indices.

5.5.1 Data Staging Plan

To support the staging activities, it is always better to normalize the source data before it is transported to the dimensional model. This may be appropriate for a particularly thorny or critical relationship. In this area, normalizing the data helps to classify and sort the source data easily (as the data will be coming from stores). The main aim of this planning is to cleanse the data which is coming from external and internal data sources.

5.5.2 Surrogate Keys

Every data table in a relational database management system typically has a unique primary key and foreign keys. However, dimensional modeling theory suggests the use of a new type of non-intelligent key, called a surrogate key and also known as a meaningless key, integer key, non-natural key, artificial key, or synthetic key [5].

Simply put, surrogate keys are integers that are assigned sequentially as needed to populate a dimension. For example, the first product record is assigned a product surrogate key with the value of 1; the next product record is assigned a product key 2, and so forth. The surrogate keys merely serve to join the dimension tables to the fact table.

One of the primary benefits of surrogate keys is that they buffer the data warehouse environment from operational changes. Surrogate keys allow the warehouse team to maintain control of the environment rather than being whipsawn by operational rules for generating, updating, deleting, recycling, and reusing production codes.
Additionally, there are performance-advantages associated with surrogate keys and these keys are needed to support one of the primary techniques for handling changes to dimension table attributes (see Chapter 4). I have defined surrogate keys for each dimension table, and typically have “_ID” as post fix (see Table 5.3). For this project, surrogate key on each table is used mainly for keeping track of historical records in that particular table.

5.6. THE SALES FORCE FACT TABLE

The following subsections give additional details dealing with the degenerate dimension and incremental loading for the sales force fact table.

5.6.1 Degenerate Dimension in Fact Table

The transaction number (a column in the sales force fact table) and transaction date act as degenerate dimensions and are useful because they serve as the grouping key for pulling together all the products purchased in a single transaction.

Although the transaction number (ID) looks like a dimension key in the fact table, I have stripped off all the descriptive items that might otherwise fall in a transaction dimension. Since the resulting dimension is empty, I refer to the transaction number and transaction date as a degenerate dimension (identified by the DD notation in Appendix B, Figure B.1). The natural operational ticket number, such as the transaction number, sits by itself in the fact table without joining a dimension table. A degenerate dimension represents a single transaction or transaction line item because the degenerate dimension represents the unique identifier of the parent.

Degenerate dimensions often play an integral role in the fact table’s primary key. In the sales database, the primary key of the sales fact table consists of the degenerate transaction_num and product key (because system rolls up all sales for a given product within a shopping cart into a single line item). Often, the primary key of a fact table is a subset of the table’s foreign keys.

Operational control numbers such as order numbers, invoice numbers, and bill-of-materials numbers usually give rise to empty dimensions. They are represented as degenerate dimensions (that is, dimension keys without corresponding dimension tables) in fact tables, where the grain of the table is the document itself or a line item in the document [5].
5.6.2 Incremental Loading into Sales Fact Table

Usually, there are several transactions done per day at each store, whenever there are any updates or insertions done by the user. As DBA, I am allowing changes and updates only to the dimension tables, since the user/guest has permissions to read from dimension tables. I am keeping the fact table unchanged till the end of the business day. Basically, I will be loading and updating the sales-force fact table once in a day only. I am using this strategy especially to avoid conflicts in reports. My fact table is the key table used for generating the reports. If I update it every time I update the dimension table, it would take a lot of time to fetch and produce reports. This can be annoying for business users and can generate reporting inconsistencies and inaccuracies. I am using the job scheduler utility in SQL Server 2005 Express to accomplish this task.

After loading data into the fact table accurately, I am creating summarized tables which contain weekly and monthly aggregated sales information for fulfilling short term reporting goals. If needed, materialized views can be generated for faster query processing. Moreover, to deal with a detailed and complicated data analysis, direct querying on the fact table could be facilitated, if required and demanded by the business people.

The above two strategies, implemented concurrently, achieve a number of operational objectives:

1. The model and the fact table are more accurate.
2. It allows for flexibility; faster processing times are possible. For example, weekly and monthly reports, can be generated with more confidence.
3. The requirements of the user are duly catered for, without the DBA losing hold on the system (the DBA is also involved in daily updating, getting the feel of the dynamics of varying data-composition and impact on the model’s performance).

5.7 PARTITIONING

The data fragmentation concept, in the context of distributed databases, aims to reduce the query execution time and facilitates the parallel execution of queries. Creating the physical storage plan for the data warehouse or data mart is not dissimilar to that for other relational databases. I, as a database administrator (DBA), need to consider the database file layout, including stripping, to minimize input-output contention.

Large fact tables are typically partitioned by date, with data segmented by month, quarter, or year into separate storage partitions, while appearing to the users as a single table.
The advantages of partitioning by date are twofold. Queries will perform better because they only access the partitions required to resolve the query. Likewise, in most cases, data loads will run faster because one (operator/DBA) only needs to rebuild the index for a partition, and not for the entire table.

Partitioning of the warehouse data is more complex and challenging compared to that stored in relational and object databases, due to the several choices of partitioning of a star schema [16]. In a data warehouse or data mart, either the dimension tables or the fact table or both can be fragmented.

Partitions can also be archived easily using SQL Server Management Studio Express 2005. Currently, quantity of the data in the data base is not considerably large. Hence, I am not doing partitioning on any table for now.

5.8 SYNONYMS

A synonym is an alternative name for a schema-scoped object. Client applications can use a single-part name to reference a base object by using a synonym instead of using a two-part, three-part, or four-part name to reference the base object. These names are usually used for convenience and require no storage other than their definition in the data dictionary. Synonyms are different words with identical or very similar meanings for any procedure, table, function, or view.

A synonym is a database object that serves the following purposes [17]:

- Provides an alternative name for another database object, referred to as the base object, which can exist on a local or remote server.
- Provides a layer of abstraction that protects a client application from changes made to the name or location of the base object.

For example, consider the Product table of Sales database, located on a server named Server1. To reference this table from another server, Server2, a client application would have to use the four-part name Server1.Sales.Product. Also, if the location of the table were to change, for example, to another server, the client application would have to be modified to reflect that change.

To address both these issues, one can create a synonym, Prod_Table, on Server2 for the Product table on Server1. Now, the client application only has to use the single-part name, Prod_Table, to reference the Product table. Also, if the location of the Product table
changes, one will have to modify the synonym, Prod_Table, to point to the new location of the Product table. Since there is no ALTER SYNONYM statement, one needs to drop the synonym, Prod_Table, and then re-create the synonym with the same name, but point the synonym to the new location of Product. Moreover, synonyms provide location transparency for remote objects in a distributed database, and simplify SQL statements for users in a (distributed) database system.

Synonyms can be easily created and maintained in the SQL server management studio 2005 (as shown in Figure 5.3). I have created a synonyms table called “product_syn” for the sales database, since this table is going to be a very common table (and since product is a conformed dimension) in the entire data warehouse picture.

Figure 5.3. Creating s synonym using SQL Server Management Studio 2005.
5.9 REPORTING TOOLS

Reporting tools are another important aspect of data mart project as reporting is one of the main reasons for building the data mart or data warehouse. There are many reporting tools available in the market currently. The level of detail, in which data is to be represented, becomes the selection criterion for the tool.

Here, I will compare some of the most popular reporting tools among developers, including tools for .NET, Java, COM and Open Source development projects. The review of these tools is based around five criteria as outlined below, and the comparisons are summarized in Figure 5.4 [18].

- **Data source support**
  What types of data sources does the tool support? Is the support limited to only the most popular database formats? Is there a data source API available for data sources when a driver is not available? In addition, does the tool support the use of stored procedures and allow advanced SQL to be used as the basis of the report?

- **Report designer**
  How well is it organized? How easy is it to learn how to use it? Further, measure the ability of the report designer to create a specific report format, how much coding was required etc.

- **Report presentation**
  It’s important to look at what types of reports can be created using this particular tool and what report formats does it support (i.e. tables, banded reports, graphs, charts, etc.). In addition, does the tool support complex reports that incorporate different sections or sub-sections, combining different report types and formats?

- **Output formats**
  Users often want to take an output report and perform further analysis, or distribute the report to other users. These criteria look at the output formats that the tool supports and how well the export and output functionalities work. Also covered here is the suitability of the exported file. For example, most tools support exporting to Excel, but does this export provide a usable spreadsheet?

- **Report integration**
  Integrating reports into existing applications should be a straightforward process and here we look at how the integration works, as well as how easy it is to get started with a particular tool or platform.
5.10 TESTING AND VALIDATION

As building a data mart is an iterative process, testing the data mart gets an extraordinary importance in the overall data mart life cycle. The main objective of testing is to ensure that all the processes execute and function with an expanded set of data. I am proposing the following steps to conduct sales data mart testing [5,10,19]:

1. Notify the business owner / sponsor and business users (or designated testers) of the available data mart test environment.
2. Test the data mart in the validation environment and verify that it meets business needs.
3. Support the data mart test environment as needed.
4. Perform constraint testing. During constraint testing, the objective is to validate unique constraints, primary keys, foreign keys, indices, and relationships.
5. Write test cases and verify the results.
6. Participate in an end-of-phase review. If there are changes to the data mart requirements, enter modifications in the appropriate dimensional model -- logical or physical data model. Changes to models must be resubmitted to the approval processes at the highest level of change.
7. Verify and measure the metadata and maintain history of the metadata tables [20, 21].

5.11 SUMMARY

The database designing process actually creates a data model and physical table components for query execution. The database software that I am using is SQL Server 2005 and Management Studio 2005 Express for designing the SIR. SQL Server 2005 provides rich security features to protect data and network resources. It is much easier to install securely, since all but the most essential features are either not installed by default or disabled if they are installed. SQL Server provides plenty of tools to configure the server. Its authentication
features are stronger because they are more closely integrated with Windows authentication and protect against weak or ancient passwords.

For SIR, I have created seven dimension tables and one fact table, and created and discussed primary and surrogate keys for each table. The main reason for using surrogate keys is to avoid embedding intelligence in the data warehouse keys, as any assumptions that are made during the construction phase of data mart eventually may be invalidated. Likewise, queries and data access applications should not have any built-in dependency on the keys because the logic would also be vulnerable to invalidation.

Different roles and user privileges are assigned for the sales database. Further strategy for updating the dimension table and loading the fact table is discussed, which is beneficial for removing conflicts and inaccuracies in the data at the time of report generation. Finally, comparison of different reporting tools is discussed, and steps for testing the sales data mart are proposed.
CHAPTER 6

FUTURE SCOPE

Ralph Kimball’s bus architecture framework, suggested for building a data warehouse in this project, is very flexible and can be extended in many ways. It can support multiple data marts and states a clear strategy for integrating these data marts with iterative planning and fixing business requirements. The SIR data model suggested for building sales data mart can also be extended to incorporate more dimensions for detailed data analysis. Future work based on this project could include:

1. Create GUI for reporting and data processing;
2. Development of data marts for other departments in Cal’Q Inc.;
3. Create detail data integration strategy development and integration;
4. Integrated reports generation. For example, integrated report based on aggregated data from sales and order management data marts;
5. Use power tools like Informatica for ETL processing;
6. Conduct a detailed testing as per the steps proposed.
REFERENCES


APPENDIX A

GLOSSARY
Ad hoc queries – Queries that are formulated by the user on the spur of the moment. The ad hoc attack refers to the difficulty a database has in anticipating the pattern of queries. The more those queries are ad hoc, the more symmetric the database model must be so that all queries look the same.

Aggregates – Physical rows in a database, almost always created by summing other records in the database for the purpose of improving query performance.

Bus – Originally used in the electrical power industry to refer to the common structure providing power; then used in the computer industry to refer to a standard interface specification. In the data warehouse, the bus refers to the standard interface that allows separate data marts to coexist usefully.

Business measure – Business performance metric captured by an operational system and represented as a fact in a dimensional model.

Business process – Major operational activities or processes supported by a source system, such as orders, from which data can be collected for the analytic purposes of the data warehouse. Choosing the business process is the first of four key steps in the design of a dimensional model.

Cube – Name for a dimensional structure on a multidimensional or online analytical processing (OLAP) database platform, originally referring to the simple three-dimension case of product, market, and time.

Data extract – Process of copying data from an operational system in order to load it into a data warehouse.

Database management system (DBMS) – A computer application whose sole purpose is to store, retrieve, and modify data in a highly structured way. Data in a DBMS usually is shared by a variety of applications.

DBA – Database administrator, a senior IT position requiring extensive understanding of database and data warehouse technology, as well as the uses of corporate data.

Dimension – An independent entity in a dimensional model that serves as an entry point or as a mechanism for slicing and dicing the additive measures located in the fact table of the dimensional model.

Dimensional data warehouse – Set of tables for decision support designed as star-joined schemas.

Entity-relationship (ER) diagram (ERD) – Drawings of boxes and lines to communicate the relationship between tables. Both third normal form (3NF) and dimensional models can be represented as ER diagrams because both consist of joined relational tables. The key difference between the models is the degree of dimension normalization. A dimensional model is a second normal form (2NF) model.
**Fact** – A business performance measurement, typically numeric and additive, that is stored in a fact table.

**Fact dimension** – A special dimension used to identify extremely sparse, dissimilar measurements in a single fact table.

**Gross profit** – The gross revenue less the cost of the goods.

**Implementation bus matrix** – A more detailed version of the data warehouse bus matrix where fact tables are identified for each business process, as well as the fact table granularity and measurements.

**Index** – A data structure associated with a table that is logically ordered by the values of a key and used to improve database performance and query access speed. B-tree indexes are used for high-cardinality fields, and bitmap indexes are used for medium- and low-cardinality fields.

**Many-to-many relationship** – A logical data relationship in which the value of one data element can exist in combination with many values of another data element, and vice versa.

**Meta data** – Any data maintained to support the operations or use of a data warehouse, similar to an encyclopedia for the data warehouse. Nearly all data staging and access tools require some private meta data in the form of specifications or status. There are few coherent standards for meta data viewed in a broader sense. Distinguished from the primary data in the dimension and fact tables.

**Natural key** – The identifier used by the operational systems. Natural keys often have embedded meaning. They may appear as dimension attributes in dimensional models but should not serve as the dimension table primary key, which always should be a surrogate key.

**Normalize** – A logical modeling technique that removes data redundancy by separating the data into many discrete entities, each of which becomes a table in a relational DBMS.

**Online transaction processing (OLTP)** – The original description for all the activities and systems associated with entering data reliably into a database. Most frequently used in reference to relational databases, although OLTP can be used generically to describe any transaction-processing environment. Contrast with *Online analytic processing*.

**Partitioned tables** – Tables (and their associated indices) that are managed as physically separate tables but appear logically as a single table. Large fact tables are candidates for partitioning, often by date. Partitioning can improve both query and maintenance performance.

**Physical design** – The phase of a database design following the logical design that identifies the actual database tables and index structures used to implement the logical design.
**Relational database management system (RDBMS)** – Database management system based on the relational model that supports the full range of standard. Uses a series of joined tables with rows and columns to organize and store data.

**Schema** – The logical or physical design of a set of database tables, indicating the relationship among the tables.

**Slice and dice** – Ability to access a data warehouse through any of its dimensions equally. Slicing and dicing is the process of separating and combining warehouse data in seemingly endless combinations.

**Transaction** – Indivisible unit of work. A transaction processing system either performs an entire transaction or it doesn’t perform any part of the transaction.
APPENDIX B

SALES INFORMATION REPOSITORY MODEL
AND SQL QUERY DETAILS
B.1 PHYSICAL DATA MODEL FOR SALES DATA MART

This section displays the physical data model in the form of Figure B.1.

Figure B.1. Physical model for sales data mart representing star schema.

B.2 SYNTAX FOR CREATING DIMENSION TABLES AND FACT TABLE

CREATE TABLE DMDCALENDER
(
    Date_ID int,
    day_of_week int,
    day_number_in_month int,
    Cal_Week_Ending_Date datetime,
);
Cal_Week_Number_in_Year int,
Cal_Month_Name varchar(20),
Cal_Month_Number_in_Year int,
Cal_YYYY_MM varchar(8),
Cal_Quarter int,
Cal_Year_Quarter varchar(10),
Cal_Half_Year char(10),
Cal_Year char(4),
Fiscal_Week int,
Fiscal_Week_Number_in_Year int,
Fiscal_Month_Name varchar(20),
Fiscal_Month_Number_in_Year int,
Fiscal_YYYY_MM char(8),
Fiscal_Quarter int,
Fiscal_Year_Quarter varchar(15),
Fiscal_Year char(4),
Holiday_flag char(1),
Weekday_flag char(3),
Last_day_in_month_flag char(1),
Selling_Season varchar(25),
Major_Event varchar(30),
Dat_Rec_Start_date datetime,
Dat_Rec_End_date datetime,
PRIMARY KEY (Date_ID, Dat_Rec_Start_date)
)

CREATE TABLE DMD_STOR_LOCATION
(
Location_ID int,
Store_code int,
City varchar(20),
Address varchar(35),
Zipcode int,
Region char(10),
State varchar(15),
Loc_Rec_Start_date datetime,
Loc_Rec_End_date datetime,
PRIMARY KEY (Location_ID, Loc_Rec_Start_date)
);

CREATE TABLE DMD_SERV_PROVIDER_INFO
(
Provider_id int,
Prov_code int,
Prov_name varchar(20),
contact_medium varchar(15),
contact_details varchar(35),
Prov_City varchar(20),
Prov_Address varchar(35),
Prov_Zipcode int,
Prov_Région char(10),
Prov_State varchar(15),
Prov_Rec_Start_date datetime,
Prov_Rec_End_date datetime,
PRIMARY KEY (Provider_ID, Prov_Rec_Start_date)
);
CREATE TABLE DMD_PRODUCT_INFO  
(  
Product_id int,  
Product_name varchar(20),  
Product_SKU int,  
Model_num varchar(10),  
Model_description varchar(35),  
Prod_Features varchar(25),  
Accessories_provided varchar(30),  
Product_color varchar(8),  
Prod_Launching_date datetime,  
Prod_Active char(1),  
Prod_Rec_Start_date datetime,  
Prod_Rec_End_date datetime,  
PRIMARY KEY (Product_ID, Prod_Rec_Start_date)  
);  

CREATE TABLE DMD_SELLING_MEDIUM  
(  
Medium_ID int,  
Medium_code varchar(12),  
Medium_desc varchar(30),  
Medium_short_desc varchar(20),  
Med_Rec_Start_date datetime,  
Med_Rec_End_date datetime,  
PRIMARY KEY (Medium_ID, Med_Rec_Start_date)  
);  

CREATE TABLE DMD_PROMOTION_INFO  
(  
Promo_ID int,  
Promo_name varchar(15),  
Promo_code int,  
Price_reduction_type varchar(15),  
Promo_media_type varchar(20),  
Ad_type char(15),  
Display_type varchar(20),  
Coupon_type varchar(15),  
AD_media_name varchar(20),  
Discount_percentage int,  
Promo_begin_date datetime,  
Promo_end_date datetime,  
Promo_Rec_Start_date datetime,  
Promo_Rec_End_date datetime,  
PRIMARY KEY (Promo_ID, Promo_Rec_Start_date)  
);  

CREATE TABLE DMD_PRICE_INFO  
(  
Price_ID int,  
Product_ID int,  
Product_price_per_unit int,  
Product_price_in_bulk int,  
Price_Effective_date datetime,  
List_price int,  

```
CREATE TABLE DMF_SALES_FACT
(
    Transaction_ID_DD int,
    Transaction_date_DD datetime,
    Date_ID int,
    Product_ID int,
    Medium_ID int,
    Provider_ID int,
    Location_ID int,
    Promo_ID int,
    Price_ID int,
    Sales_Quantity int,
    Dollar_sales_Amount int,
    Cost_Dollar_amount int,
    Fact_Rec_Start_Date datetime,
    Fact_Rec_End_Date datetime,
    Loc_Rec_Start_Date datetime,
    Promo_Rec_Start_Date datetime,
    Prov_Rec_Start_Date datetime,
    Med_Rec_Start_Date datetime,
    Pric_Rec_Start_Date datetime,
    Dat_Rec_Start_Date datetime,
    Prod_Rec_Start_Date datetime
);

B.3 DESCRIPTIVE FIGURES

Figures B.2 and B.3 show visual descriptions of the screens for assigning privileges and scheduling jobs using Microsoft SQL Server Management Studio Express.
Figure B.2. Assigning privileges to respective roles using Microsoft SQL Server Management Studio Express.
Figure B.3. Scheduling job using Microsoft SQL Server Management Studio Express.