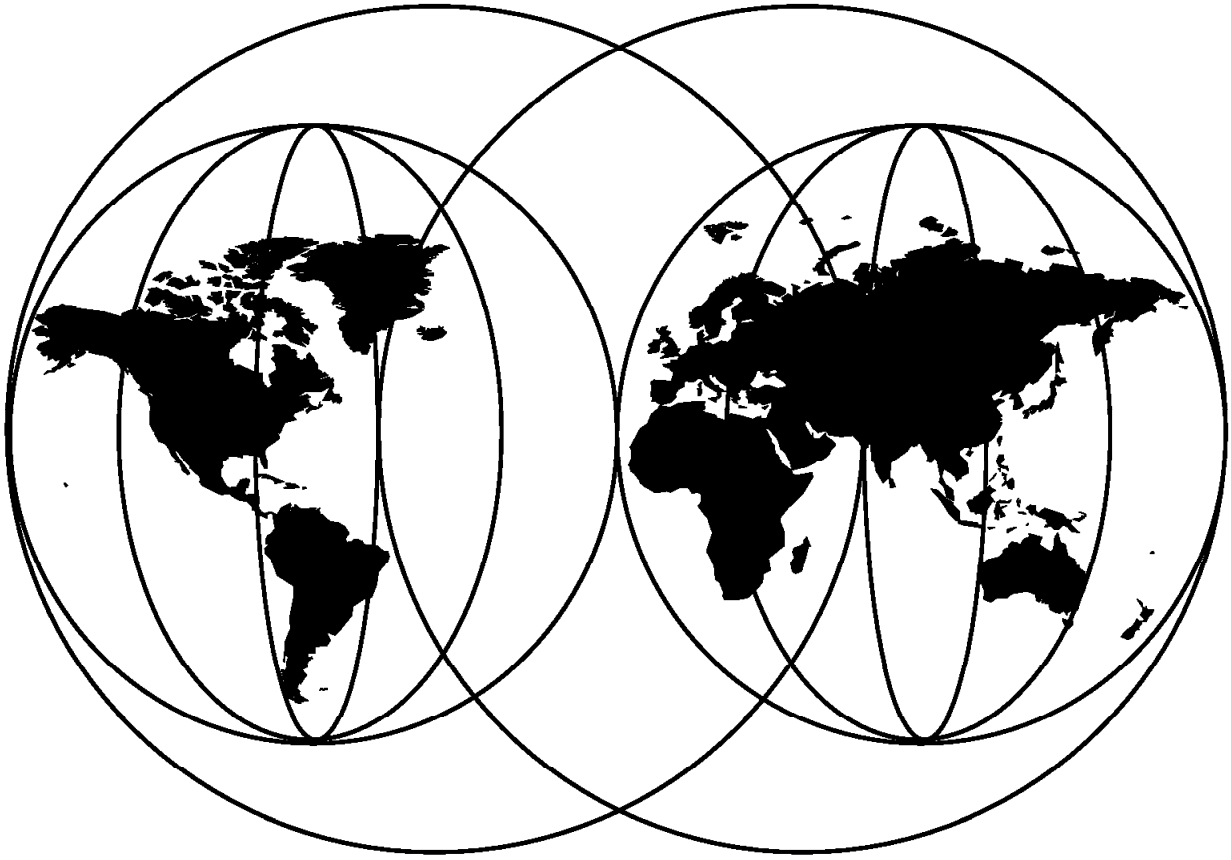
Data Modeling Techniques for Data Warehousing

Chuck Ballard, Dirk Herreman, Don Schau, Rhonda Bell,
Eunsaeng Kim, Ann Valencic



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-2238-00

International Technical Support Organization

SG24-2238-00

Data Modeling Techniques for Data Warehousing

February 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special Notices” on page 183.

First Edition (February 1998)

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099

Contents

Figures

Tables

Preface

This redbook gives detail coverage to the topic of data modeling techniques for data warehousing, within the context of the overall data warehouse development process. The process of data warehouse modeling, including the steps required before and after the actual modeling step, is discussed. Detailed coverage of modeling techniques is presented in an evolutionary way through a gradual, but well-managed, expansion of the content of the actual data model. Coverage is also given to other important aspects of data warehousing that affect, or are affected by, the modeling process. These include architecting the warehouse and populating the data warehouse. Guidelines for selecting a data modeling tool that is appropriate for data warehousing are presented.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working for the IBM International Technical Support Organization San Jose center.

Chuck Ballard was the project manager for the development of the book and is currently a data warehousing consultant at the IBM International Technical Support Organization-San Jose center. He develops, staffs, and manages projects to explore current topics in data warehousing that result in the delivery of technical workshops, papers, and IBM Redbooks. Chuck writes extensively and lectures worldwide on the subject of data warehousing. Before joining the ITSO, he worked at the IBM Santa Teresa Development Lab, where he was responsible for developing strategies, programs, and market support deliverables on data warehousing.

Dirk Herreman is a senior data warehousing consultant for CIMAD Consultants in Belgium. He leads a team of data warehouse consultants, data warehouse modelers, and data and system architects for data warehousing and operates with CIMAD Consultants within IBM′s Global Services. Dirk has more than 15 years of experience with databases, most of it from an application development point of view. For the last couple of years in particular, his work has focused primarily on the development of process and architecture models and the associated techniques for evolutionary data warehouse development. As a result of this work, Dirk and his team are now the prime developers of course and workshop materials for IBM′s worldwide education curriculum for data warehouse enablement. He holds a degree in mathematics and in computer sciences from the State University of Ghent, Belgium.

Don Schau is an Information Consultant for the City of Winnipeg. He holds a diploma in analysis and programming from Red River Community College. He has 20 years of experience in data processing, the last 8 in data and database management, with a focus on data warehousing in the past 2 years. His areas of expertise include data modeling and data and database management. Don currently resides in Winnipeg, Manitoba, Canada with his wife, Shelley, and their four children.

Rhonda Bell is an I/T Architect in the Business Intelligence Services Practice for IBM Global Services based in Austin, Texas. She has 5 years of experience in data processing. Rhonda holds a degree in computer information systems from

Southwest Texas State University. Her areas of expertise include data modeling and client/server and data warehouse design and development.

Eunsaeng Kim is an Advisory Sales Specialist in Banking, Finance and Securities Industry (BFSI) for IBM Korea. He has seven years of experience in data processing, the last five years in banking data warehouse modeling and implementation for four Korean commercial banks. He holds a degree in economics from Seoul National University in Seoul, Korea. His areas of expertise include data modeling, data warehousing, and business subjects in banking and finance industry. Eunsaeng currently resides in Seoul, Korea with his wife, Eunkyung and their two sons.

Ann Valencic is a Senior Systems Specialist in the Software Services Group in IBM Australia. She has 12 years of experience in data processing, specializing in database and data warehouse. Ann′s areas of expertise include database design and performance tuning.

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 197 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com`
  For IBM Intranet users `http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@vnet.ibm.com`

Chapter 1. Introduction

Businesses of all sizes and in different industries, as well as government agencies, are finding that they can realize significant benefits by implementing a data warehouse. It is generally accepted that data warehousing provides an excellent approach for transforming the vast amounts of data that exist in these organizations into useful and reliable information for getting answers to their questions and to support the decision making process. A data warehouse provides the base for the powerful data analysis techniques that are available today such as data mining and multidimensional analysis, as well as the more traditional query and reporting. Making use of these techniques along with data warehousing can result in easier access to the information you need for more informed decision making.

The question most asked now is, How do I build a data warehouse? This is a question that is not so easy to answer. As you will see in this book, there are many approaches to building one. However, at the end of all the research, planning, and architecting, you will come to realize that it all starts with a firm foundation. Whether you are building a large centralized data warehouse, one or more smaller distributed data warehouses (sometimes called data marts), or some combination of the two, you will always come to the point where you must decide on how the data is to be structured. This is, after all, one of the most key concepts in data warehousing and what differentiates it from the more typical operational database and decision support application building. That is, you structure the data and build applications around it rather than structuring applications and bringing data to them.

How will you structure the data in your data warehouse? The purpose of this book is to help you with that decision. It all revolves around data modeling. Everyone will have to develop a data model; the decision is how much effort to expend on the task and what type of data model should be used. There are new data modeling techniques that have become popular in recent years and provide excellent support for data warehousing. This book discusses those techniques and offers some considerations for their selection in a data warehousing environment.

Data warehouse modeling is a process that produces abstract data models for one or more database components of the data warehouse. It is one part of the overall data warehouse development process, which is comprised of other major processes such as data warehouse architecture, design, and construction. We consider the data warehouse modeling process to consist of all tasks related to requirements gathering, analysis, validation, and modeling. Typically for data warehouse development, these tasks are difficult to separate. The book covers data warehouse design only at a superficial level. This may suggest a rather broad gap between modeling and design activities, which in reality certainly is not the case. The separation between modeling and design is done for practical reasons: it is our intention to cover the modeling activities and techniques quite extensively. Therefore, covering data warehouse design as extensively simply could not be done within the scope of this book.

The need to model data warehouse databases in a way that differs from modeling operational databases has been promoted in many textbooks. Some trend-setting authors and data warehouse consultants have taken this point to what we consider to be the extreme. That is, they are presenting what they are

calling a totally new approach to data modeling. It is called *dimensional* data modeling, or *fact/dimension* modeling. Fancy names have been invented to refer to different types of dimensional models, such as star models and snowflake models. Numerous arguments have been presented against traditional entity-relationship (ER) modeling, when used for modeling data in the data warehouse. Rather than taking this more extreme position, we believe that every technique has its area of usability. For example, we do support the many criticisms of ER modeling when considered in a specific context of data warehouse data modeling, and there are also criticisms of dimensional modeling. There are many types of data warehouse applications for which ER modeling is not well suited, especially those that address the needs of a well-identified community of data analysts interested primarily in analyzing *their* business measures in *their* business context. Likewise, there are data warehouse applications that are not well supported at all by star or snowflake models alone. For example, dimensional modeling is not very suitable for making large, corporatewide data models for a data warehouse.

With the changing data warehouse landscape and the need for data warehouse modeling, the new modeling approaches and the controversies surrounding traditional modeling and the dimensional modeling approach all merit investigation. And that is another purpose of this book. Because it presents details of data warehouse modeling processes and techniques, the book can also be used as an initiating textbook for those who want to learn data warehouse modeling.

1.1 Who Should Read This Book

This book is intended for those involved in the development, implementation, maintenance, and administration of data warehouses. It is also applicable for project planners and managers involved in data warehousing.

To benefit from this book, the reader should have, at least, a basic understanding of ER modeling.

It is worthwhile for those responsible for developing a data warehouse to progress sequentially through the entire book. Those less directly involved in data warehouse modeling should refer to 1.2, “Structure of This Book” to determine which chapters will be of interest.

1.2 Structure of This Book

In Chapter 2, “Data Warehousing” on page 5, we begin with an exploration of the evolution of the concept of data warehousing, as it relates to data modeling for the data warehouse. We discuss the subject of data marts and distinguish them from data warehouses. After having read Chapter 1, you should have a clear perception of data modeling in the context of data mart and/or data warehouse development.

Chapter 3, “Data Analysis Techniques” on page 9 surveys several methods of data analysis in data warehousing. Query and reporting, multidimensional analysis, and data mining run the spectrum of being analyst driven to analyst assisted to data driven. Because of this spectrum, each of the data analysis methods affects data modeling.

Chapter 4, “Data Warehousing Architecture and Implementation Choices” on page 15 discusses the architecture and implementation choices available for data warehousing. The architecture of the data warehouse environment is based on where the data warehouses and/or data marts reside and where the control of the data exists. Three architecture choices are presented: the global warehouse, independent data marts, and interconnected data marts. There are several ways to implement these architecture choices: top down, bottom up, or stand alone. These three implementation choices offer flexibility in choosing an architecture and deploying the resources to create the data warehouse and/or data marts within the organization.

Chapter 5, “Architecting the Data” on page 23 addresses the approaches and techniques suitable for architecting the data in the data warehouse. Information requirements can be satisfied by three types of business data: real-time, reconciled, and derived. The Enterprise Data Model (EDM) could be very helpful in data warehouse data modeling, if you have one. For example, from the EDM you could derive the general scope and understanding of the business requirements, and you could link the EDM to the physical area of interest. Also discussed in this chapter is the importance of data granularity, or level of detail of the data.

Chapter 6, “Data Modeling for a Data Warehouse” on page 35 presents the basics of data modeling for the data warehouse. Two major approaches are described. First we present the highlights of ER modeling, identify the major components of ER models, and describe their properties. Next, we introduce the basic concepts of dimensional modeling and present and position two fundamental approaches: Star modeling and Snowflake. We also position the different approaches by contrasting ER and dimensional modeling, and Stars and Snowflakes. We also identify how and when the different approaches can be used as complementary, and how the different models and techniques can be mapped.

In Chapter 7, “The Process of Data Warehousing” on page 49, we present a process model for data warehouse modeling. This is one of the core chapters of this book. Data modeling techniques are covered extensively in Chapter 8, “Data Warehouse Modeling Techniques” on page 81, but they can only be appreciated and well used if they are part of a well-managed data warehouse modeling process. The process model we use as the base for this book is an evolutionary, user-centric approach. It is one that focuses on end-user requirements first (rather than on the data sources) and recognizes that data warehouses and data marts typically are developed with a bottom-up approach.

Chapter 8, “Data Warehouse Modeling Techniques” on page 81 covers the core data modeling techniques for the data warehouse development process. The chapter has two major sections. In the first section, we present the techniques suitable for developing a data warehouse or a data mart that suits the needs of a particular community of end users or data analysts. In the second section, we explore the data warehouse modeling techniques suitable for expanding the scope of a data mart or a data warehouse. The techniques presented in this chapter are of particular interest for those organizations that develop their data marts or data warehouses in an evolutionary way; that is, through a gradual, but well-managed, expansion of the scope of content of what has already been implemented.

Chapter 9, “Selecting a Modeling Tool” on page 155, an overview of the functions that a data modeling tool, or suite of tools, must support for modeling the data warehouse is presented. Also presented is a partial list of tools available at the time this redbook was written.

Chapter 10, “Populating the Data Warehouse” on page 159 discusses the process of populating the data warehouse or data mart. Populating is the process of getting the source data from the operational and external systems into the data warehouse and data marts. This process consists of a capture step, a transform step, and an apply step. Also discussed in this chapter is the effect of modeling on the populating process, and, conversely, the effect of populating on modeling.

Chapter 2. Data Warehousing

In this chapter we position data warehousing as more than just a product, or set of products—it is a solution! It is an information environment that is separate from the more typical transaction-oriented operational environment. Data warehousing is, in and of itself, an information environment that is evolving as a critical resource in today's organizations.

2.1 A Solution, Not a Product

Often we think that a data warehouse is a product, or group of products, that we can buy to help get answers to our questions and improve our decision-making capability. But, it is not so simple. A data warehouse can help us get answers for better decision making, but it is only one part of a more global set of processes. As examples, where did the data in the data warehouse come from? How did it get into the data warehouse? How is it maintained? How is the data structured in the data warehouse? What is actually in the data warehouse? These are all questions that must be answered before a data warehouse can be built. We prefer to discuss the more global environment, and we refer to it as data warehousing.

Data warehousing is the design and implementation of processes, tools, and facilities to manage and deliver complete, timely, accurate, and understandable information for decision making. It includes all the activities that make it possible for an organization to create, manage, and maintain a data warehouse or data mart.

2.2 Why Data Warehousing?

The concept of data warehousing has evolved out of the need for easy access to a structured store of quality data that can be used for decision making. It is globally accepted that information is a very powerful asset that can provide significant benefits to any organization and a competitive advantage in the business world. Organizations have vast amounts of data but have found it increasingly difficult to access it and make use of it. This is because it is in many different formats, exists on many different platforms, and resides in many different file and database structures developed by different vendors. Thus organizations have had to write and maintain perhaps hundreds of programs that are used to extract, prepare, and consolidate data for use by many different applications for analysis and reporting. Also, decision makers often want to dig deeper into the data once initial findings are made. This would typically require modification of the extract programs or development of new ones. This process is costly, inefficient, and very time consuming. Data warehousing offers a better approach.

Data warehousing implements the process to access heterogeneous data sources; clean, filter, and transform the data; and store the data in a structure that is easy to access, understand, and use. The data is then used for query, reporting, and data analysis. As such, the access, use, technology, and performance requirements are completely different from those in a transaction-oriented operational environment. The volume of data in data warehousing can be very high, particularly when considering the requirements

for historical data analysis. Data analysis programs are often required to scan vast amounts of that data, which could result in a negative impact on operational applications, which are more performance sensitive. Therefore, there is a requirement to separate the two environments to minimize conflicts and degradation of performance in the operational environment.

2.3 Short History

The origin of the concept of data warehousing can be traced back to the early 1980s, when relational database management systems emerged as commercial products. The foundation of the relational model with its simplicity, together with the query capabilities provided by the SQL language, supported the growing interest in what then was called *end-user computing* or *decision support*. To support end-user computing environments, data was extracted from the organization′s online databases and stored in newly created database systems dedicated to supporting ad hoc end-user queries and reporting functions of all kinds. One of the prime concerns underlying the creation of these systems was the performance impact of end-user computing on the operational data processing systems. This concern prompted the requirement to separate end-user computing systems from transactional processing systems.

In those early days of data warehousing, the extracts of operational data were usually snapshots or subsets of the operational data. These snapshots were loaded in an end-user computing (or decision support) database system on a regular basis, perhaps once a week or once per month. Sometimes a limited number of versions of these snapshots were even accumulated in the system while access was provided to end users equipped with query and reporting tools. Data modeling for these decision support database systems was not much of a concern. Data models for these decision support systems typically matched the data models of the operational systems because, after all, they were extracted snapshots anyhow. One of the frequently occurring remodeling issues then was to ″normalize″ the data to eliminate the nasty effects of design techniques that had been applied on the operational systems to maximize their performance, to eliminate code tables that were difficult to understand, along with other local cleanup activities. But by and large, the decision support data models were technical in nature and primarily concerned with providing data available in the operational application systems to the decision support environment.

The role and purpose of data warehouses in the data processing industry have evolved considerably since those early days and are still evolving rapidly. Comparing today′s data warehouses with the early days′ decision support databases should be done with great care. Data warehouses should no longer be identified with database systems that support end-user queries and reporting functions. They should no longer be conceived as snapshots of operational data. Data warehouse databases should be considered as new sources of information, conceived for use by the whole organization or for smaller communities of users and data analysts within the organization. Simply reengineering source data models in the traditional way will no longer satisfy the requirements for data warehousing. Developing data warehouses requires a much more thoughtfully applied set of modeling techniques and a much closer working relationship with the business side of the organization.

Data warehouses should also be conceived of as sources of new information. This statement sounds controversial at first, because there is global agreement that data warehouses are read-only database systems. The point is, that by

accumulating and consolidating data from different sources, and by keeping this historical data in the warehouse, new information about the business, competitors, customers, suppliers, the behavior of the organization′s business processes, and so forth, can be unveiled. The value of a data warehouse is no longer in being able to do ad hoc query and reporting. The real value is realized when someone gets to work with the data in the warehouse and discovers things that make a difference for the organization, whatever the objective of the analytical work may be. To achieve such interesting results, simply reengineering the source data models will not do.

Chapter 3. Data Analysis Techniques

A data warehouse is built to provide an easy to access source of high quality data. It is a means to an end, not the end itself. That end is typically the need to perform analysis and decision making through the use of that source of data. There are several techniques for data analysis that are in common use today. They are query and reporting, multidimensional analysis, and data mining (see Figure 1). They are used to formulate and *display* query results, to *analyze* data content by viewing it from different perspectives, and to *discover* patterns and clustering attributes in the data that will provide further insight into the data content.

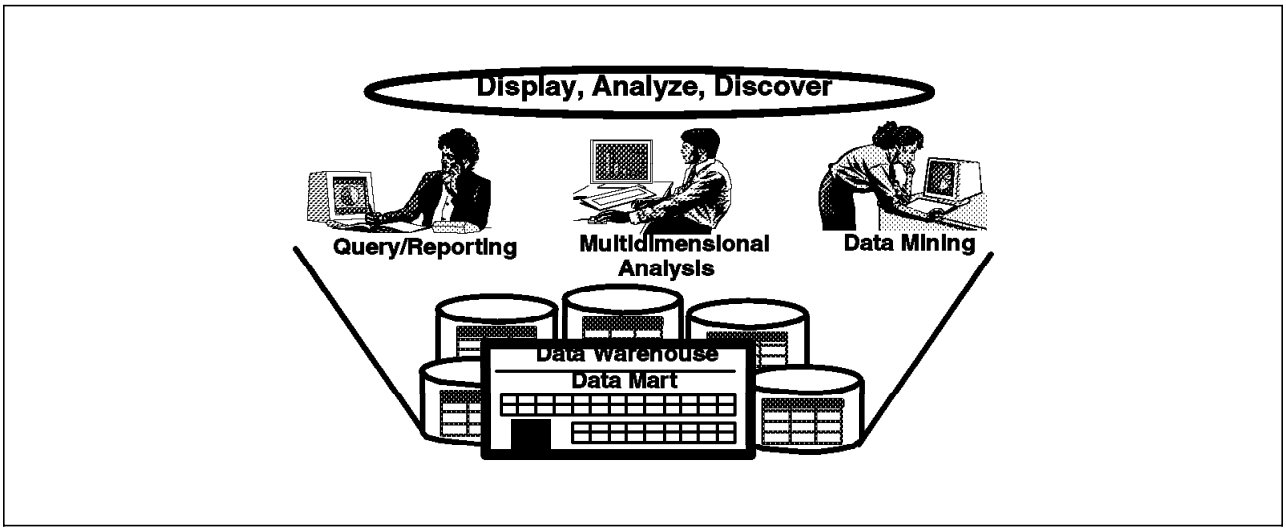


Figure 1. Data Analysis. Several methods of data analysis are in common use.

The techniques of data analysis can impact the type of data model selected and its content. For example, if the intent is simply to provide query and reporting capability, a data model that structures the data in more of a normalized fashion would probably provide the fastest and easiest access to the data. Query and reporting capability primarily consists of selecting associated data elements, perhaps summarizing them and grouping them by some category, and presenting the results. Executing this type of capability typically might lead to the use of more direct table scans. For this type of capability, perhaps an ER model with a normalized and/or denormalized data structure would be most appropriate.

If the objective is to perform multidimensional data analysis, a dimensional data model would be more appropriate. This type of analysis requires that the data model support a structure that enables fast and easy access to the data on the basis of any of numerous combinations of analysis dimensions. For example, you may want to know how many of a specific product were sold on a specific day, in a specific store, in a specific price range. Then for further analysis you may want to know how many stores sold a specific product, in a specific price range, on a specific day. These two questions require similar information, but one viewed from a product perspective and the other viewed from a store perspective.

Multidimensional analysis requires a data model that will enable the data to easily and quickly be viewed from many possible perspectives, or dimensions.

Since a number of dimensions are being used, the model must provide a way for fast access to the data. If a highly normalized data structure were used, many joins would be required between the tables holding the different dimension data, and they could significantly impact performance. In this case, a dimensional data model would be most appropriate.

An understanding of the data and its use will impact the choice of a data model. It also seems clear that, in most implementations, multiple types of data models might be used to best satisfy the varying requirements of the data warehouse.

3.1 Query and Reporting

Query and reporting analysis is the process of posing a question to be answered, retrieving relevant data from the data warehouse, transforming it into the appropriate context, and displaying it in a readable format. It is driven by analysts who must pose those questions to receive an answer. You will find that this is quite different, for example, from data mining, which is data driven. Refer to Figure 4 on page 13.

Traditionally, queries have dealt with two dimensions, or two factors, at a time. For example, one might ask, ″How much of that product has been sold this week?″ Subsequent queries would then be posed to perhaps determine how much of the product was sold by a particular store. Figure 2 depicts the process flow in query and reporting. Query definition is the process of taking a business question or hypothesis and translating it into a query format that can be used by a particular decision support tool. When the query is executed, the tool generates the appropriate language commands to access and retrieve the requested data, which is returned in what is typically called an *answer set*. The data analyst then performs the required calculations and manipulations on the answer set to achieve the desired results. Those results are then formatted to fit into a display or report template that has been selected for ease of understanding by the end user. This template could consist of combinations of text, graphic images, video, and audio. Finally, the report is delivered to the end user on the desired output medium, which could be printed on paper, visualized on a computer display device, or presented audibly.

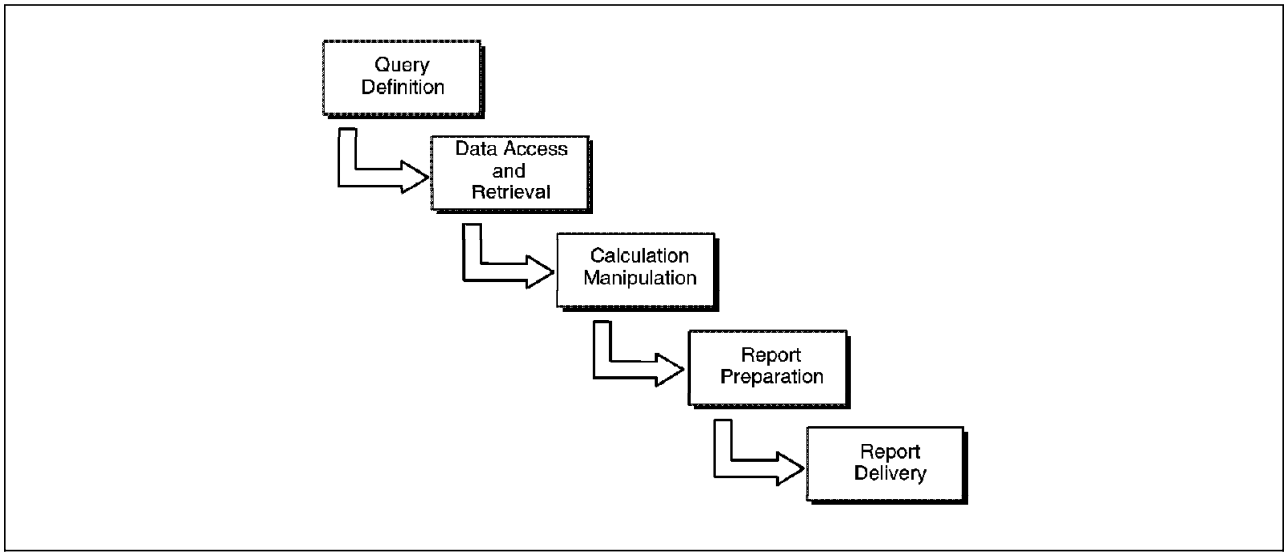


Figure 2. Query and Reporting. The process of query and reporting starts with query definition and ends with report delivery.

End users are primarily interested in processing numeric values, which they use to analyze the behavior of business processes, such as sales revenue and shipment quantities. They may also calculate, or investigate, quality measures such as customer satisfaction rates, delays in the business processes, and late or wrong shipments. They might also analyze the effects of business transactions or events, analyze trends, or extrapolate their predictions for the future. Often the data displayed will cause the user to formulate another query to clarify the answer set or gather more detailed information. This process continues until the desired results are reached.

3.2 Multidimensional Analysis

Multidimensional analysis has become a popular way to extend the capabilities of query and reporting. That is, rather than submitting multiple queries, data is structured to enable fast and easy access to answers to the questions that are typically asked. For example, the data would be structured to include answers to the question, ″How much of each of our products was sold on a particular day, by a particular sales person, in a particular store?″ Each separate part of that query is called a *dimension*. By precalculating answers to each subquery within the larger context, many answers can be readily available because the results are not recalculated with each query; they are simply accessed and displayed. For example, by having the results to the above query, one would automatically have the answer to any of the subqueries. That is, we would already know the answer to the subquery, ″How much of a particular product was sold by a particular salesperson?″ Having the data categorized by these different factors, or dimensions, makes it easier to understand, particularly by business-oriented users of the data. Dimensions can have individual entities or a hierarchy of entities, such as region, store, and department.

Multidimensional analysis enables users to look at a large number of interdependent factors involved in a business problem and to view the data in complex relationships. End users are interested in exploring the data at different levels of detail, which is determined dynamically. The complex relationships can be analyzed through an iterative process that includes drilling down to lower levels of detail or rolling up to higher levels of summarization and aggregation. Figure 3 on page 12 demonstrates that the user can start by viewing the total sales for the organization and drill down to view the sales by continent, region, country, and finally by customer. Or, the user could start at customer and roll up through the different levels to finally reach total sales. Pivoting in the data can also be used. This is a data analysis operation whereby the user takes a different viewpoint than is typical on the results of the analysis, changing the way the dimensions are arranged in the result. Like query and reporting, multidimensional analysis continues until no more drilling down or rolling up is performed.

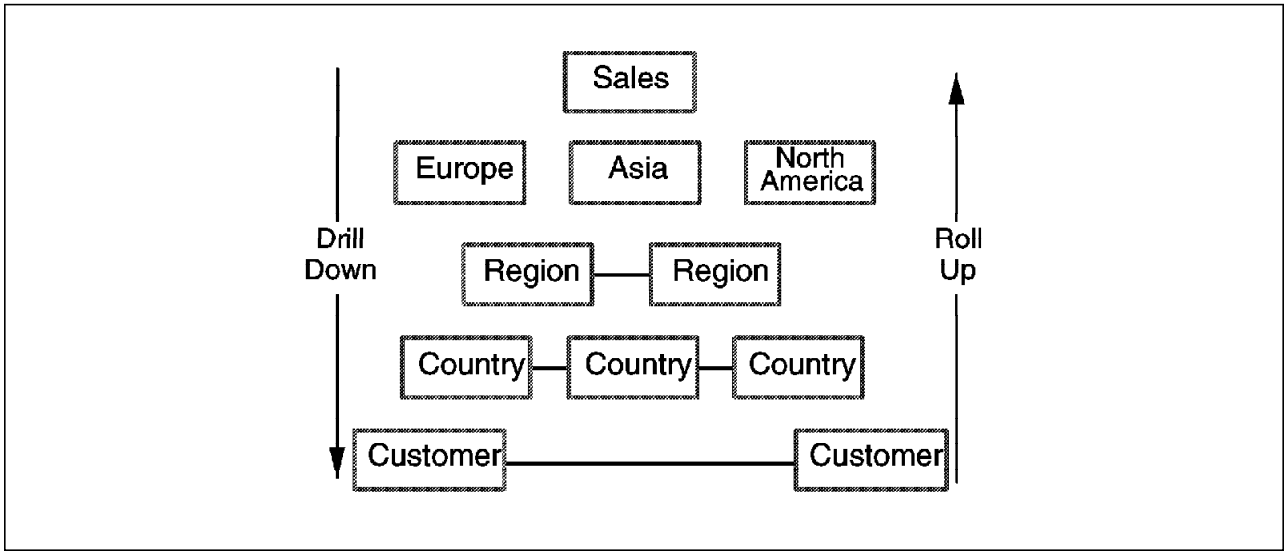


Figure 3. Drill-Down and Roll-Up Analysis. End users can perform drill down or roll up when using multidimensional analysis.

3.3 Data Mining

Data mining is a relatively new data analysis technique. It is very different from query and reporting and multidimensional analysis in that is uses what is called a *discovery technique*. That is, you do not ask a particular question of the data but rather use specific algorithms that analyze the data and report what they have discovered. Unlike query and reporting and multidimensional analysis where the user has to create and execute queries based on hypotheses, data mining searches for answers to questions that may have not been previously asked. This discovery could take the form of finding significance in relationships between certain data elements, a clustering together of specific data elements, or other patterns in the usage of specific sets of data elements. After finding these patterns, the algorithms can infer rules. These rules can then be used to generate a model that can predict a desired behavior, identify relationships among the data, discover patterns, and group clusters of records with similar attributes.

Data mining is most typically used for statistical data analysis and knowledge discovery. Statistical data analysis detects unusual patterns in data and applies statistical and mathematical modeling techniques to explain the patterns. The models are then used to forecast and predict. Types of statistical data analysis techniques include linear and nonlinear analysis, regression analysis, multivariant analysis, and time series analysis. Knowledge discovery extracts implicit, previously unknown information from the data. This often results in uncovering unknown business facts.

Data mining is data driven (see Figure 4 on page 13). There is a high level of complexity in stored data and data interrelations in the data warehouse that are difficult to discover without data mining. Data mining offers new insights into the business that may not be discovered with query and reporting or multidimensional analysis. Data mining can help discover new insights about the business by giving us answers to questions we might never have thought to ask.

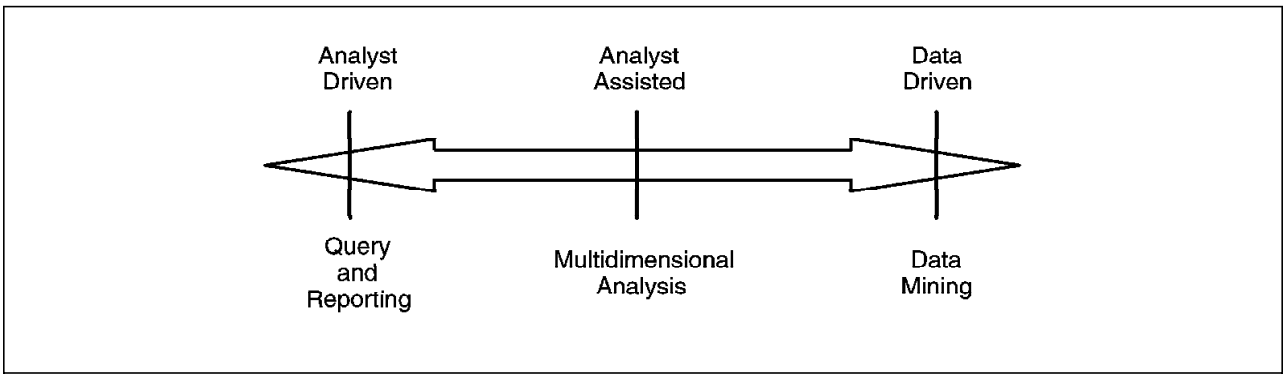


Figure 4. Data Mining. Data Mining focuses on analyzing the data content rather than simply responding to questions.

3.4 Importance to Modeling

The type of analysis that will be done with the data warehouse can determine the type of model and the model's contents. Because query and reporting and multidimensional analysis require summarization and explicit metadata, it is important that the model contain these elements. Also, multidimensional analysis usually entails drilling down and rolling up, so these characteristics need to be in the model as well. A clean and clear data warehouse model is a requirement, else the end users' tasks will become too complex, and end users will stop trusting the contents of the data warehouse and the information drawn from it because of highly inconsistent results.

Data mining, however, usually works best with the lowest level of detail available. Thus, if the data warehouse is used for data mining, a low level of detail data should be included in the model.

Chapter 4. Data Warehousing Architecture and Implementation Choices

In this chapter we discuss the architecture and implementation choices available for data warehousing. During the discussions we may use the term *data mart*. Data marts, simply defined, are smaller data warehouses that can function independently or can be interconnected to form a global integrated data warehouse. However, in this book, unless noted otherwise, use of the term *data warehouse* also implies data mart.

Although it is not always the case, choosing an architecture should be done prior to beginning implementation. The architecture can be determined, or modified, after implementation begins. However, a longer delay typically means an increased volume of rework. And, everyone knows that it is more time consuming and difficult to do rework after the fact than to do it right, or very close to right, the first time. The architecture choice selected is a management decision that will be based on such factors as the current infrastructure, business environment, desired management and control structure, commitment to and scope of the implementation effort, capability of the technical environment the organization employs, and resources available.

The implementation approach selected is also a management decision, and one that can have a dramatic impact on the success of a data warehousing project. The variables affected by that choice are time to completion, return-on-investment, speed of benefit realization, user satisfaction, potential implementation rework, resource requirements needed at any point-in-time, and the data warehouse architecture selected.

4.1 Architecture Choices

Selection of an architecture will determine, or be determined by, where the data warehouses and/or data marts themselves will reside and where the control resides. For example, the data can reside in a central location that is managed centrally. Or, the data can reside in distributed local and/or remote locations that are either managed centrally or independently.

The architecture choices we consider in this book are global, independent, interconnected, or some combination of all three. The implementation choices to be considered are top down, bottom up, or a combination of both. It should be understood that the architecture choices and the implementation choices can also be used in combinations. For example, a data warehouse architecture could be physically distributed, managed centrally, and implemented from the bottom up starting with data marts that service a particular workgroup, department, or line of business.

4.1.1 Global Warehouse Architecture

A global data warehouse is considered one that will support all, or a large part, of the corporation that has the requirement for a more fully integrated data warehouse with a high degree of data access and usage across departments or lines-of-business. That is, it is designed and constructed based on the needs of the enterprise as a whole. It could be considered to be a common repository for

decision support data that is available across the entire organization, or a large subset thereof.

A common misconception is that a global data warehouse is centralized. The term *global* is used here to reflect the scope of data access and usage, not the physical structure. The global data warehouse can be physically centralized or physically distributed throughout the organization. A physically centralized global warehouse is to be used by the entire organization that resides in a single location and is managed by the Information Systems (IS) department. A distributed global warehouse is also to be used by the entire organization, but it distributes the data across multiple physical locations within the organization and is managed by the IS department.

When we say that the IS department manages the data warehouse, we do not necessarily mean that it *controls* the data warehouse. For example, the distributed locations could be controlled by a particular department or line of business. That is, they decide what data goes into the data warehouse, when it is updated, which other departments or lines of business can access it, which individuals in those departments can access it, and so forth. However, to manage the implementation of these choices requires support in a more global context, and that support would typically be provided by IS. For example, IS would typically manage network connections. Figure 5 shows the two ways that a global warehouse can be implemented. In the top part of the figure, you see that the data warehouse is distributed across three physical locations. In the bottom part of the figure, the data warehouse resides in a single, centralized location.

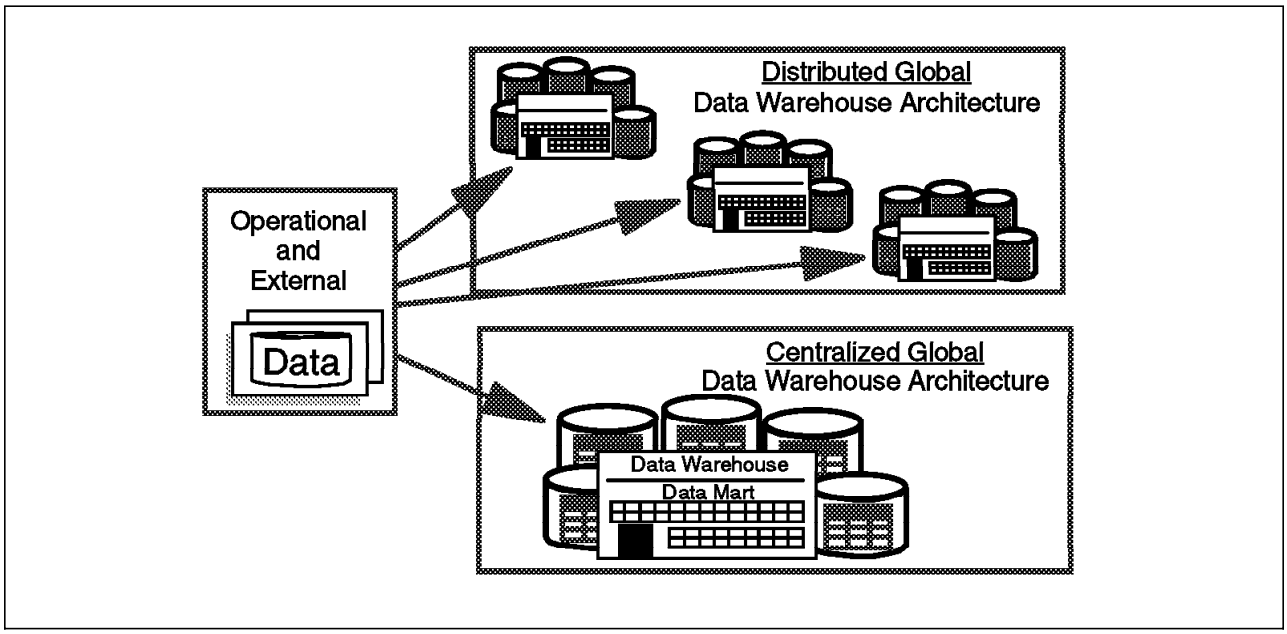


Figure 5. Global Warehouse Architecture. The two primary architecture approaches.

Data for the data warehouse is typically extracted from operational systems and possibly from data sources external to the organization with batch processes during off-peak operational hours. It is then filtered to eliminate any unwanted data items and transformed to meet the data quality and usability requirements. It is then loaded into the appropriate data warehouse databases for access by end users.

A global warehouse architecture enables end users to have more of an enterprisewide or corporatewide view of the data. It should be certain that this is a requirement, however, because this type of environment can be very time consuming and costly to implement.

4.1.2 Independent Data Mart Architecture

An independent data mart architecture implies stand-alone data marts that are controlled by a particular workgroup, department, or line of business and are built solely to meet their needs. There may, in fact, not even be any connectivity with data marts in other workgroups, departments, or lines of business. For example, data for these data marts may be generated internally. The data may be extracted from operational systems but would then require the support of IS. IS would not control the implementation but would simply help manage the environment. Data could also be extracted from sources of data external to the organization. In this case IS could be involved unless the appropriate skills were available within the workgroup, department, or line of business. The top part of Figure 6 depicts the independent data mart structure. Although the figure depicts the data coming from operational or external data sources, it could also come from a global data warehouse if one exists.

The independent data mart architecture requires some technical skills to implement, but the resources and personnel could be owned and managed by the workgroup, department, or line of business. These types of implementation typically have minimal impact on IS resources and can result in a very fast implementation. However, the minimal integration and lack of a more global view of the data can be a constraint. That is, the data in any particular data mart will be accessible only to those in the workgroup, department, or line of business that owns the data mart. Be sure that this is a known and accepted situation.

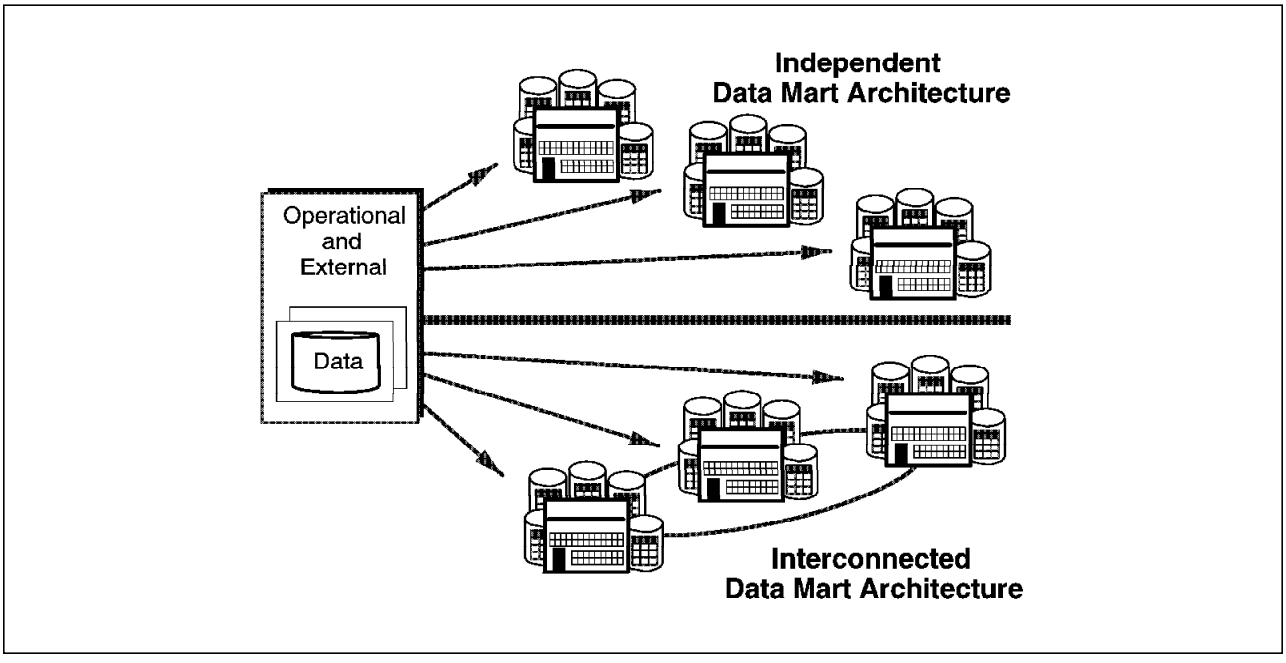


Figure 6. Data Mart Architectures. They can be independent or interconnected.

4.1.3 Interconnected Data Mart Architecture

An interconnected data mart architecture is basically a distributed implementation. Although separate data marts are implemented in a particular workgroup, department, or line of business, they can be integrated, or interconnected, to provide a more enterprisewide or corporatewide view of the data. In fact, at the highest level of integration, they can become the global data warehouse. Therefore, end users in one department can access and use the data on a data mart in another department. This architecture is depicted in the bottom of Figure 6 on page 17. Although the figure depicts the data coming from operational or external data sources, it could also come from a global data warehouse if one exists.

This architecture brings with it many other functions and capabilities that can be selected. Be aware, however, that these additional choices can bring with them additional integration requirements and complexity as compared to the independent data mart architecture. For example, you will now need to consider who controls and manages the environment. You will need to consider the need for another tier in the architecture to contain, for example, data common to multiple data marts. Or, you may need to elect a data sharing schema across the data marts. Either of these choices adds a degree of complexity to the architecture. But, on the positive side, there can be significant benefit to the more global view of the data.

Interconnected data marts can be independently controlled by a workgroup, department, or line of business. They decide what source data to load into the data mart, when to update it, who can access it, and where it resides. They may also elect to provide the tools and skills necessary to implement the data mart themselves. In this case, minimal resources would be required from IS. IS could, for example, provide help in cross-department security, backup and recovery, and the network connectivity aspects of the implementation. In contrast, interconnected data marts could be controlled and managed by IS. Each workgroup, department, or line of business would have its own data mart, but the tools, skills, and resources necessary to implement the data marts would be provided by IS.

4.2 Implementation Choices

Several approaches can be used to implement the architectures discussed in 4.1, “Architecture Choices” on page 15. The approaches to be discussed in this book are top down, bottom up, or a combination of both. These implementation choices offer flexibility in determining the criteria that are important in any particular implementation.

The choice of an implementation approach is influenced by such factors as the current IS infrastructure, resources available, the architecture selected, scope of the implementation, the need for more global data access across the organization, return-on-investment requirements, and speed of implementation.

4.2.1 Top Down Implementation

A top down implementation requires more planning and design work to be completed at the beginning of the project. This brings with it the need to involve people from each of the workgroups, departments, or lines of business that will be participating in the data warehouse implementation. Decisions concerning data sources to be used, security, data structure, data quality, data standards, and an overall data model will typically need to be completed before actual implementation begins. The top down implementation can also imply more of a need for an enterprisewide or corporatewide data warehouse with a higher degree of cross workgroup, department, or line of business access to the data. This approach is depicted in Figure 7. As shown, with this approach, it is more typical to structure a global data warehouse. If data marts are included in the configuration, they are typically built afterward. And, they are more typically populated from the global data warehouse rather than directly from the operational or external data sources.

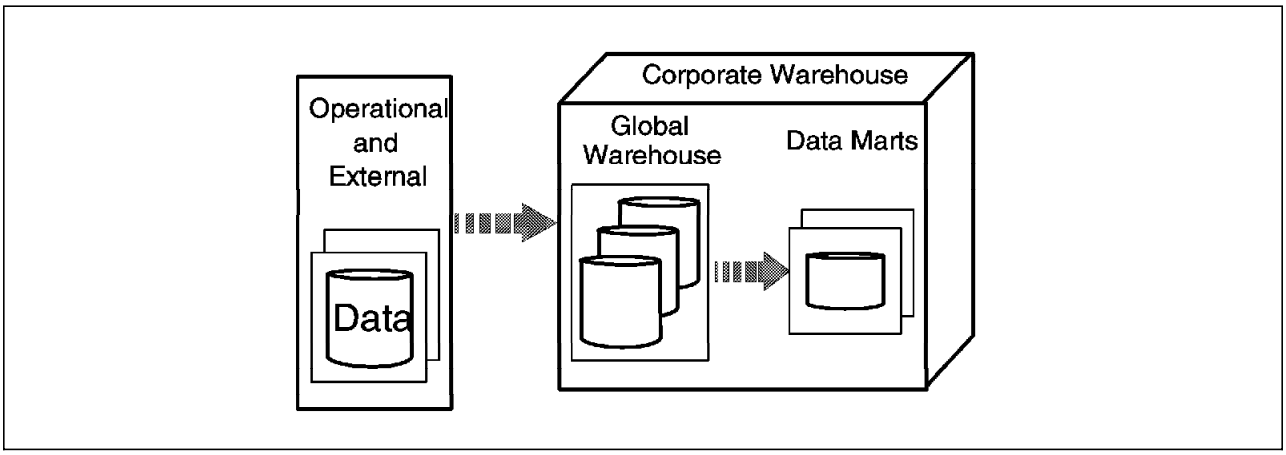


Figure 7. Top Down Implementation. Creating a corporate infrastructure first.

A top down implementation can result in more consistent data definitions and the enforcement of business rules across the organization, from the beginning. However, the cost of the initial planning and design can be significant. It is a time-consuming process and can delay actual implementation, benefits, and return-on-investment. For example, it is difficult and time consuming to determine, and get agreement on, the data definitions and business rules among all the different workgroups, departments, and lines of business participating. Developing a global data model is also a lengthy task. In many organizations, management is becoming less and less willing to accept these delays.

The top down implementation approach can work well when there is a good centralized IS organization that is responsible for all hardware and other computer resources. In many organizations, the workgroups, departments, or lines of business may not have the resources to implement their own data marts. Top down implementation will also be difficult to implement in organizations where the workgroup, department, or line of business has its own IS resources. They are typically unwilling to wait for a more global infrastructure to be put in place.

4.2.2 Bottom Up Implementation

A bottom up implementation involves the planning and designing of data marts without waiting for a more global infrastructure to be put in place. This does not mean that a more global infrastructure will not be developed; it will be built incrementally as initial data mart implementations expand. This approach is more widely accepted today than the top down approach because immediate results from the data marts can be realized and used as justification for expanding to a more global implementation. Figure 8 depicts the bottom up approach. In contrast to the top down approach, data marts can be built before, or in parallel with, a global data warehouse. And as the figure shows, data marts can be populated either from a global data warehouse or directly from the operational or external data sources.

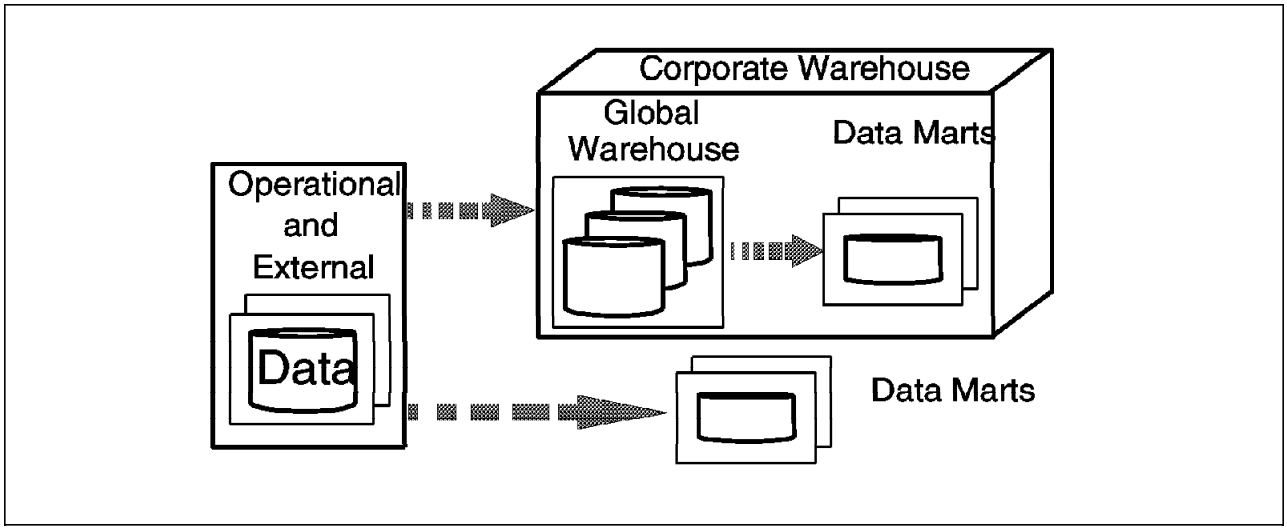


Figure 8. Bottom Up Implementation. Starts with a data mart and expands over time.

The bottom up implementation approach has become the choice of many organizations, especially business management, because of the faster payback. It enables faster results because data marts have a less complex design than a global data warehouse. In addition, the initial implementation is usually less expensive in terms of hardware and other resources than deploying the global data warehouse.

Along with the positive aspects of the bottom up approach are some considerations. For example, as more data marts are created, data redundancy and inconsistency between the data marts can occur. With careful planning, monitoring, and design guidelines, this can be minimized. Multiple data marts may bring with them an increased load on operational systems because more data extract operations are required. Integration of the data marts into a more global environment, if that is the desire, can be difficult unless some degree of planning has been done. Some rework may also be required as the implementation grows and new issues are uncovered that force a change to the existing areas of the implementation. These are all considerations to be carefully understood before selecting the bottom up approach.

4.2.3 A Combined Approach

As we have seen, there are both positive and negative considerations when implementing with the top down or the bottom up approach. In many cases the best approach may be a combination of the two. This can be a difficult balancing act, but with a good project manager it can be done. One of the keys to this approach is to determine the degree of planning and design that is required for the global approach to support integration as the data marts are being built with the bottom up approach. Develop a base level infrastructure definition for the global data warehouse, being careful to stay, initially, at a business level. For example, as a first step simply identify the lines of business that will be participating. A high level view of the business processes and data areas of interest to them will provide the elements for a plan for implementation of the data marts.

As data marts are implemented, develop a plan for how to handle the data elements that are needed by multiple data marts. This could be the start of a more global data warehouse structure or simply a common data store accessible by all the data marts. It some cases it may be appropriate to duplicate the data across multiple data marts. This is a trade-off decision between storage space, ease of access, and the impact of data redundancy along with the requirement to keep the data in the multiple data marts at the same level of consistency.

There are many issues to be resolved in any data warehousing implementation. Using the combined approach can enable resolution of these issues as they are encountered, and in the smaller scope of a data mart rather than a global data warehouse. Careful monitoring of the implementation processes and management of the issues could result in gaining the best benefits of both implementation techniques.

Chapter 5. Architecting the Data

A data warehouse is, by definition, a subject-oriented, integrated, time-variant collection of *data* to enable decision making across a disparate group of users. One of the most basic concepts of data warehousing is to clean, filter, transform, summarize, and aggregate the data, and then put it in a structure for easy access and analysis by those users. But, that structure must first be defined and that is the task of the data warehouse model. In modeling a data warehouse, we begin by architecting the data. By architecting the data, we structure and locate it according to its characteristics.

In this chapter, we review the types of data used in data warehousing and provide some basic hints and tips for architecting that data. We then discuss approaches to developing a data warehouse data model along with some of the considerations.

Having an enterprise data model (EDM) available would be very helpful, but not required, in developing the data warehouse data model. For example, from the EDM you can derive the general scope and understanding of the business requirements. The EDM would also let you relate the data elements and the physical design to a specific area of interest.

Data granularity is one of the most important criteria in architecting the data. On one hand, having data of a high granularity can support any query. However, having a large volume of data that must be manipulated and managed could be an issue as it would impact response times. On the other hand, having data of a low granularity would support only specific queries. But, with the reduced volume of data, you would realize significant improvements in performance.

The size of a data warehouse varies, but they are typically quite large. This is especially true as you consider the impact of storing volumes of historical data. To deal with this issue you have to consider data partitioning in the data architecture. We consider both logical and physical partitioning to better understand and maintain the data. In logical partitioning of data, you should consider the concept of *subject areas*. This concept is typically used in most information engineering (IE) methodologies. We discuss subject areas and their different definitions in more detail later in this chapter.

5.1 Structuring the Data

In structuring the data, for data warehousing, we can distinguish three basic types of data that can be used to satisfy the requirements of an organization:

- Real-time data
- Derived data
- Reconciled data

In this section, we describe these three types of data according to usage, scope, and currency. You can configure an appropriate data warehouse based on these three data types, with consideration for the requirements of any particular implementation effort. Depending on the nature of the operational systems, the type of business, and the number of users that access the data warehouse, you

can combine the three types of data to create the most appropriate architecture for the data warehouse.

5.1.1 Real-Time Data

Real-time data represents the current status of the business. It is typically used by operational applications to run the business and is constantly changing as operational transactions are processed. Real-time data is at a detailed level, meaning high granularity, and is usually accessed in read/write mode by the operational transactions.

Not confined to operational systems, real-time data is extracted and distributed to informational systems throughout the organization. For example, in the banking industry, where real-time data is critical for operational management and tactical decision making, an independent system, the so-called *deferred* or *delayed* system, delivers the data from the operational systems to the informational systems (data warehouses) for data analysis and more strategic decision making.

To use real-time data in a data warehouse, typically it first must be cleansed to ensure appropriate data quality, perhaps summarized, and transformed into a format more easily understood and manipulated by business analysts. This is because the real-time data contains all the individual, transactional, and detailed data values as well as other data valuable only to the operational systems that must be filtered out. In addition, because it may come from multiple different systems, real-time data may not be consistent in representation and meaning. As an example, the units of measure, currency, and exchange rates may differ among systems. These anomalies must be reconciled before loading into the data warehouse.

5.1.2 Derived Data

Derived data is data that has been created perhaps by summarizing, averaging, or aggregating the real-time data through some process. Derived data can be either detailed or summarized, based on requirements. It can represent a view of the business at a specific point in time or be a historical record of the business over some period of time.

Derived data is traditionally used for data analysis and decision making. Data analysts seldom need large volumes of detailed data; rather they need summaries that are much easier for manipulation and use. Manipulating large volumes of atomic data can also require tremendous processing resources. Considering the requirements for improved query processing capability, an efficient approach is to precalculate derived data elements and summarize the detailed data to better meet user requirements. Efficiently processing large volumes of data in an appropriate amount of time is one of the most important issues to resolve.

5.1.3 Reconciled Data

Reconciled data is real-time data that has been cleansed, adjusted, or enhanced to provide an integrated source of quality data that can be used by data analysts. The basic requirement for data quality is consistency. In addition, we can create and maintain historical data while reconciling the data. Thus, we can say reconciled data is a special type of derived data.

Reconciled data is seldom explicitly defined. It is usually a logical result of derivation operations. Sometimes reconciled data is stored only as temporary files that are required to transform operational data for consistency.

5.2 Enterprise Data Model

An EDM is a consistent definition of all of the data elements common to the business, from a high-level business view to a generic logical data design. It includes links to the physical data designs of individual applications. Through an EDM, you can derive the general scope and understanding of the business requirements.

5.2.1 Phased Enterprise Data Modeling

Many methodologies for enterprise data modeling have been published. Some publications propose a three-tiered methodology such as conceptual, logical, and physical data model. In IBM′s Worldwide Solution Design and Delivery Method (WSDDM), five tiers are described for information engineering approaches:

- ISP - Information System Planning
- BAA - Business Area Analysis
- BSD - Business System Design
- BSI - Business System Implementation
- BSM - Business System Maintenance

Despite the differences of the number of tiers, the common thread is that every methodology focuses on the *phased* or layered approach. The phases include the tasks for information planning, business analyzing, logical data modeling, and physical data design as shown on Figure 9.

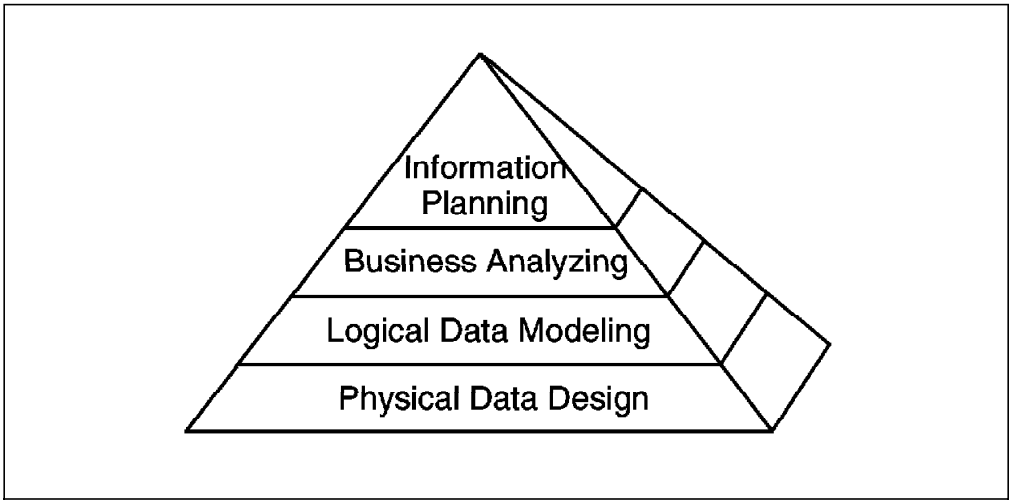


Figure 9. The Phased Enterprise Data Model (EDM)

The size of the phases in Figure 9 represents the amount of information to be included in that phase of the model. That is, the pyramid shape implies that the amount of information is minimal at the planning phase but increases remarkably at the physical data design phase.

The *information planning* phase at the top of the pyramid provides the highly consolidated view of the business. In that view, we can identify some number of business concepts, usually in the range of 10 to 30. Those business concepts

can be categorized as a *business entity*, *super entity*, or *subject area* in which an organization is interested and about which it maintains data elements. Examples of those are customer, product, organization, schedule, activity, and policy. The purpose of this phase is to set up the *scope and architecture* of a data warehouse and to provide a single, comprehensive point of view for the other phases.

The *business analyzing* phase provides a means of further defining the contents of the primary concepts and categorizing those contents according to various business rules. This phase is described in business terms so that business people who have no modeling training can understand it. The purpose of this phase is to gather and arrange business requirements and define the business terms specifically.

The *logical data modeling* phase is primarily enterprisewide in scope and generic to all applications located below it in the pyramid. The logical data model typically consists of several hundred entities. It is a complete model that is in third normal form and contains the identification and definition of all entities, relationships, and attributes. For further specific analysis, the entities of the logical data model are sometimes partitioned into views by subject areas or by applications. Some methodologies divide this phase into two phases:

- Generic logical data model - the enterprise level
- Logical application model - application level of data view

The *physical data design* applies physical constraints, such as space, performance, and the physical distribution of data. The purpose of this phase is to design for the actual physical implementation.

5.2.2 A Simple Enterprise Data Model

In general it is not possible, or practical, to assign resources to all of the development phases concurrently when constructing an EDM. However, some of the core components that are required for data warehouse modeling can be extracted and grouped and used as a phased approach. In this book we call that phased approach a *simple EDM*.

Figure 10 on page 27 shows an example of a simple EDM that consists of subject areas and the relationships among them. We suggest drawing a simple EDM diagram for each subject you select for your data warehouse model.

For a simple EDM, make a list of subject areas, typically less than 25. Then, draw a *subject area model* of the organization by defining the business relationships among the subject areas.

When you have completed the subject area model, you will need to define the contents of each subject area. For example, when you define *customer*, you cannot simply say that customer is the person in an organization that has a business relationship with your organization. For example, you must make it clear whether the *person* includes a prospect or ex-customer. When referring to the *organization*, be clear as to whether it can be only a registered business, and not simply a social or civic interest group.

If possible, draw an ER diagram for each subject area. Do not be too concerned about the details, such as relationship name and cardinality. Just identify the primary entities and the relationships among them.

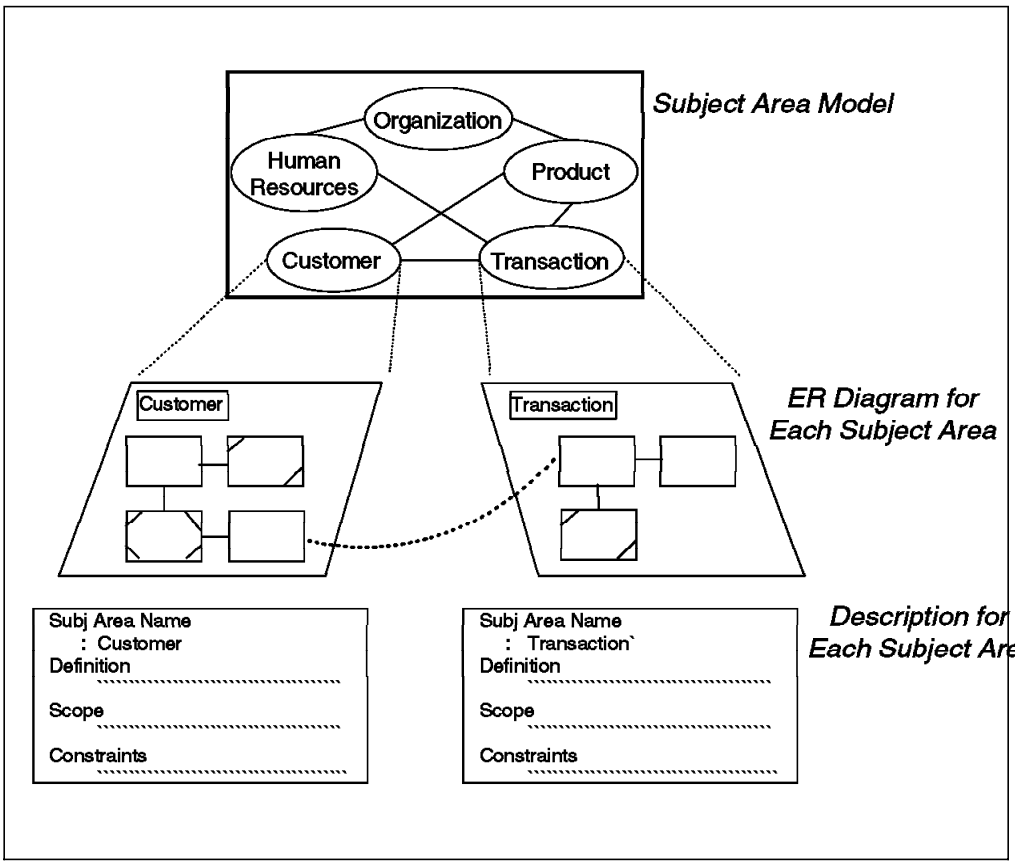


Figure 10. A Simple Enterprise Data Model

The objective of a simple EDM is to scope a specific area of interest and develop a general understanding of it. It will be very helpful in the development of your data warehouse model.

5.2.3 The Benefits of EDM

Compared to an application or departmental model, an EDM has these benefits:

- Provides a single development base for applications and promotes the integration of existing applications
- Supports the sharing of data among different areas of the business
- Provides a single set of consistent data definitions

The benefits of the EDM are being challenged today because a number of organizations have attempted to create them and have been largely unsuccessful. The following are some of the reasons for this lack of success:

- The scope of EDM tends to cover the entire enterprise. Therefore, the size of the project tended to be so big that it seldom delivered the proper results in a reasonable period of time.
- To deliver the value of EDM to the business, all areas of the organization must be involved concurrently, which is an unrealistic expectation.
- The people required in an EDM project must have both a broad understanding of the business and a detailed knowledge of a specific business area. It is difficult to find such people, but even if you can, it is more difficult to get them assigned to the modeling task rather than performing their standard business duties.

The above reasons are certainly cause for concern, but we consider them challenges rather than reasons to avoid pursuit of an EDM. It is still a valuable item to have and can be very helpful in creating the data warehouse model. To help ease the effort of creating an EDM, many industry-specific template data models are available to use as a starting point. For example, there is the Financial Services Data Model (FSDM) for the finance industry available from IBM. Through customizing the templates, you can reduce the modeling period and required resources while at the same time experience the stable benefits of an EDM.

If an organization has no EDM and no plans to create one, you can still receive many of the benefits by creating a simple EDM. Whether the scope of the data warehouse is for the entire enterprise or for a specific business area, a simple EDM adds value. If you already have several data models for specific applications, you can make use of them in creating the simple EDM. For example, you can extract common components from application data models and integrate them into the simple EDM. Integration is always a virtue in data warehousing.

5.3 Data Granularity Model

In the physical design phase for data modeling, one of the most important aspects of the design is related to the *granularity* of the data. In this section we describe what we mean by granularity in the context of a data warehouse and explain how to structure data to minimize or eliminate any loss of information from using this valuable construct.

5.3.1 Granularity of Data in the Data Warehouse

Granularity of data in the data warehouse is concerned with the *level of summarization* of the data elements. It refers then, actually, to the level of *detail* available in the data elements. The more detail data that is available, the lower the level of granularity. Conversely, the lower the level of detail, the higher the level of granularity (or level of summarization of the data elements).

Granularity is important in data warehouse modeling because it offers the opportunity for trade-off between important issues in data warehousing. For example, one trade-off could be performance versus volume of data (and the related cost of storing that data). Another example might be a trade-off between the ability to access data at a very detailed level versus performance and the cost of storing and accessing large volumes of data. Selecting the appropriate level of granularity significantly affects the volume of data in the data warehouse. Along with that, selecting the appropriate level of granularity determines the capability of the data warehouse to enable answers to different types of queries. To help make this clear, refer to the example shown in Figure 11 on page 29. Here we are looking at transaction data for a bank account. On the left side of the figure, let′s say that 50 is the average number of transaction per account and the size of the record for a transaction is 150 bytes. As the result, it would require about 7.5 KB to keep the very detailed transaction records to the end of the month. On the right side of the figure, a less detailed set of data (with a higher level of granularity) is shown in the form of summary by account per month. Here, all the transactions for an account are summarized in only one record. The summary record would require longer record size, perhaps 200 bytes instead of the 150 bytes of the raw transaction, but the result is a significant savings in storage space.

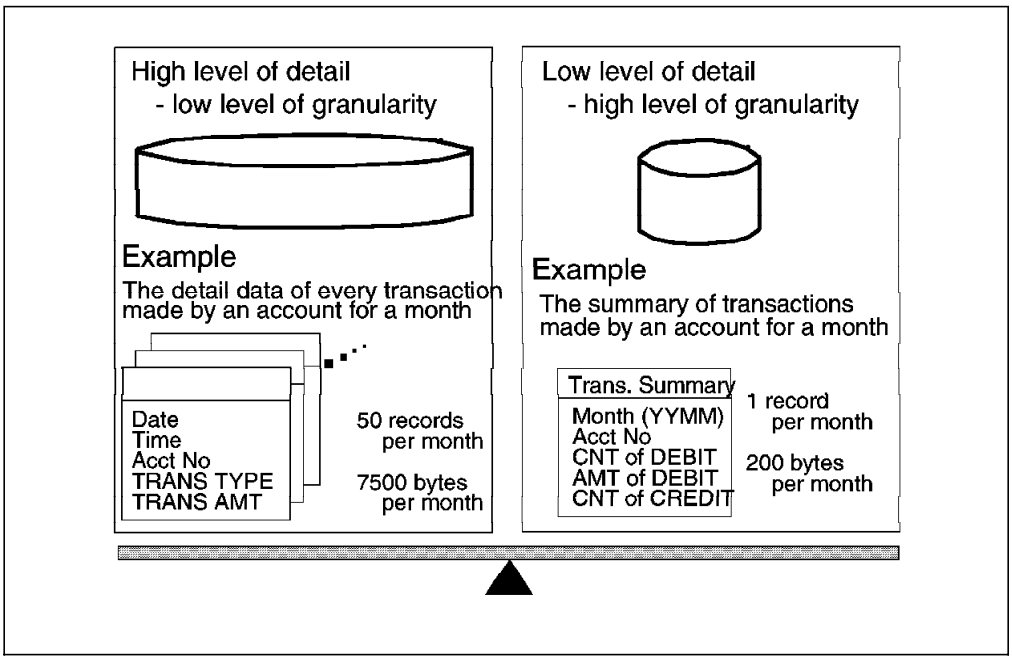


Figure 11. Granularity of Data:. The Level of Detail Trade-off

In terms of disk space and volume of data, a higher granularity provides a more efficient way of storing data than a lower granularity. You would also have to consider the disk space for the index of the data as well. This makes the space savings even greater. Perhaps a greater concern is with the manipulation of large volumes of data. This can impact performance at the cost of more processing power.

There are always trade-offs to be made in data processing, and this is no exception. For example, as the granularity becomes higher, the ability to answer different types of queries (that require data at a more detailed level) diminishes. If you have very low level of granularity, you can support any queries using that data at the cost of increased storage space and diminished performance.

Let's look again at the example in Figure 11. With a low level of granularity you could answer the query, "How many credit transactions were there for John's demand deposit account in the San Jose branch last week?" With the higher level of granularity, you cannot answer that question because the data is summarized by month rather than by week.

If the granularity does not impact the ability to answer a specific query, the amount of system resources required for that same query could still differ considerably. Suppose that you have two tables with different levels of granularity, such as transaction details and monthly account summary. To answer a query about the monthly report for channel utilization by accounts, you could use either of those two tables without any dependency on the level of granularity. However, using the detailed transaction table requires a significantly higher volume of disk activity to scan all the data as well as additional processing power for calculation of the results. Using the monthly account summary table would require much less resource.

In deciding on the level of granularity, you must always consider the trade-off between the cost of the volume of data and the ability to answer queries.

5.3.2 Multigranularity Modeling in the Corporate Environment

In organizations that have large volumes of data, multiple levels of granularity could be considered to overcome the trade-offs. For example, we could divide the data in a data warehouse into *detailed raw* data and *summarized* data.

Detailed raw data is the lowest level of detailed transaction data without any aggregation and summarization. At this level, the data volume could be extremely large. It may actually have to be on a separate storage medium such as magnetic tape or an optical disk device when it is not being used. The data could be loaded to disk for easy and faster access only during those times when it is required.

Summarized data is transaction data aggregated at the level required for the most typically used queries. In the banking example used previously, this might be at the level of customer accounts. A much lower volume of data is required for the summarized data source as compared to the detailed raw data. Of course, there is a limit to the number of queries and level of detail that can be extracted from the summarized data.

By creating two levels of granularity in a data warehouse, you can overcome the trade-off between volume of data and query capability. The summarized level of data supports almost all queries with the reduced amount of resources, and the detailed raw data supports the limited number of queries requiring a detailed level of data.

What we mean by *summarized* may still not be clear. The issue here is about what the criteria will be for determining the level of summarization that should be used in various situations. The answer requires a certain amount of intuition and experience in the business. For example, if you summarize the data at a very low level of detail, there will be few differences from the detailed raw data. If you summarize the data at too high a level of detail, many queries must be satisfied by using the detailed raw data. Therefore, in the beginning, simply using intuition may be the rule. Then, over time, analytical *iterative* processes can be refined to enhance or verify the intuition. Collecting statistics on the usage of the various sources of data will provide input for the processes.

By structuring the data into multiple levels of summarized data, you can extend the analysis of *dual* levels of granularity into *multiple* levels of granularity based on the business requirements and the capacity of the data warehouse of each organization. You will find more detail and examples of techniques for implementing multigranularity modeling in Chapter 8, “Data Warehouse Modeling Techniques” on page 81.

5.4 Logical Data Partitioning Model

To better understand, maintain, and navigate the data warehouse, we can define both logical and physical partitions. Physical partitioning can be designed according to the physical implementation requirements and constraints. In data warehouse modeling, logical data partitioning is very important because it affects physical partitioning not only for overall structure but also detailed table partitioning. In this section we describe why and how the data is partitioned.

The subject area is the most common criterion for determining overall logical data partitioning. We can define a subject area as a portion of a data warehouse that is classified by a specific consistent perspective. The perspective is usually

based on the characteristics of the data, such as customer, product, or account. Sometimes, however, other criteria such as time period, geography, and organizational unit become the measure for partitioning.

5.4.1 Partitioning the Data

The term *partition* was originally concerned with the physical status of a data structure that has been divided into two or more separate structures. However, sometimes *logical partitioning* of the data is required to better understand and use the data. In that case, the descriptions of logical partitioning overlap with physical partitioning.

5.4.1.1 The Goals of Partitioning

Partitioning the data in the data warehouse enables the accomplishment of several critical goals. For example, it can:

- Provide flexible access to data
- Provide easy and efficient data management services
- Ensure scalability of the data warehouse
- Enable elements of the data warehouse to be portable. That is, certain elements of the data warehouse can be shared with other physical warehouses or archived on other storage media.

We usually partition large volumes of current detail data by splitting it into smaller pieces. Doing that helps make the data easier to:

- Restructure
- Index
- Sequentially scan
- Reorganize
- Recover
- Monitor

5.4.1.2 The Criteria of Partitioning

For the question of how to partition the data in a data warehouse, there are a number of important criteria to consider. As examples, the data can be partitioned according to several of the following criteria:

- Time period (date, month, or quarter)
- Geography (location)
- Product (more generically, by line of business)
- Organizational unit
- A combination of the above

The choice of criteria is based on the business requirements and physical database constraints. Nevertheless, time period must always be considered when you decide to partition data.

Every database management system (DBMS) has its own specific way of implementing physical partitioning, and they all can be quite different. And, a very important consideration when selecting the DBMS on which the data resides is support for partition indexing. Instead of DBMS or system level of partitioning, you can consider partitioning by application. This would provide flexibility in defining data over time, and portability in moving to the other data warehouses. Notice that the issue of partitioning is closely related to

multidimensional modeling, data granularity modeling, and the capabilities of a particular DBMS to support data warehousing.

5.4.2 Subject Area

When you consider the partitioning of the data in a data warehouse, the most common criterion is subject area. As you will remember, a data warehouse is subject oriented; that is, it is oriented to specific selected subject areas in the organization such as customer and product. This is quite different from partitioning in the operational environment.

In the operational environment, partitioning is more typically by application or function because the operational environment has been built around transaction-oriented applications that perform a specific set of functions. And, typically, the objective is to perform those functions as quickly as possible. If there are queries performed in the operational environment, they are more tactical in nature and are to answer a question concerned with that instant in time. An example might be, ″Is the check for Mr. Smith payable or not?″ Queries in the data warehouse environment are more strategic in nature and are to answer questions concerned with a larger scope. An example might be ″What products are selling well?″ or ″Where are my weakest sales offices?″ To answer those questions, the data warehouse should be structured and oriented to subject areas such as product or organization. As such, subject areas are the most common unit of logical partitioning in the data warehouse.

Subject areas are roughly classified by the topics of interest to the business. To extract a candidate list of potential subject areas, you should first consider what your business interests are. Examples are customers, profit, sales, organizations, and products. To help in determining the subject areas, you could use a technique that has been successful for many organizations, namely, the *5W1H rule*; that is, the *when, where, who, what, why,* and *how* of your business interests. For example, for answering the *who* question, your business interests might be in customer, employee, manager, supplier, business partner, and competitor.

After you extract a list of candidate subject areas, you decompose, rearrange, select, and redefine them more clearly. As a result, you can get a list of subject areas that best represent your organization. We suggest that you make a hierarchy or grouping with them to provide a clear definition of what they are and how they relate to each other. As a practical example of subject areas, consider the following list taken from the FSDM:

- Arrangement
- Business direction item
- Classification
- Condition
- Event
- Involved party
- Location
- Product
- Resource item

The above list of nine subject areas can be decomposed into several other subject areas. For example, arrangement consists of several subject areas such as customer arrangement, facility arrangement, and security arrangement.

Once you have a list of subject areas, you have to define the business relationships among them. The relationships are good starting points for determining the dimensions that might be used in a dimensional data warehouse model because a subject area is a perspective of the business about which you are interested.

In data warehouse modeling, subject areas help define the following criteria:

- Unit of the data model
- Unit of an implementation project
- Unit of management of the data
- Basis for the integration of multiple implementations

Assuming that the main role of subject area is the determination of the unit for effective analysis, modeling, and implementation of the data warehouse, then the other criteria such as business function, process, specific applications, or organizational unit can be the measure for the subject area.

In dimensional modeling, the best unit of analysis is the business process area in which the organization has the most interest. For a practical implementation of a data warehouse, it is suggested that the unit of measure be the business process area.

Chapter 6. Data Modeling for a Data Warehouse

This chapter provides you with a basic understanding of data modeling, specifically for the purpose of implementing a data warehouse.

Data warehousing has become generally accepted as the best approach for providing an integrated, consistent source of data for use in data analysis and business decision making. However, data warehousing can present complex issues and require significant time and resources to implement. This is especially true when implementing on a corporatewide basis. To receive benefits faster, the implementation approach of choice has become bottom up with data marts. Implementing in these small increments of small scope provides a larger return-on-investment in a short amount of time. Implementing data marts does not preclude the implementation of a global data warehouse. It has been shown that data marts can scale up or be integrated to provide a global data warehouse solution for an organization. Whether you approach data warehousing from a global perspective or begin by implementing data marts, the benefits from data warehousing are significant.

The question then becomes, How should the data warehouse databases be designed to best support the needs of the data warehouse users? Answering that question is the task of the data modeler. Data modeling is, by necessity, part of every data processing task, and data warehousing is no exception. As we discuss this topic, unless otherwise specified, the term *data warehouse* also implies *data mart*.

We consider two basic data modeling techniques in this book: ER modeling and dimensional modeling. In the operational environment, the ER modeling technique has been the technique of choice. With the advent of data warehousing, the requirement has emerged for a technique that supports a data analysis environment. Although ER models can be used to support a data warehouse environment, there is now an increased interest in dimensional modeling for that task.

In this chapter, we review why data modeling is important for data warehousing. Then we describe the basic concepts and characteristics of ER modeling and dimensional modeling.

6.1 Why Data Modeling Is Important

Visualization of the business world: Generally speaking, a model is an abstraction and reflection of the real world. Modeling gives us the ability to visualize what we cannot yet realize. It is the same with data modeling.

Traditionally, data modelers have made use of the ER diagram, developed as part of the data modeling process, as a communication media with the business end users. The ER diagram is a tool that can help in the analysis of business requirements and in the design of the resulting data structure. Dimensional modeling gives us an improved capability to visualize the very abstract questions that the business end users are required to answer. Utilizing dimensional modeling, end users can easily understand and navigate the data structure and fully exploit the data.

Actually, data is simply a record of all business activities, resources, and results of the organization. The data model is a well-organized abstraction of that data. So, it is quite natural that the data model has become the best method to understand and manage the business of the organization. Without a data model, it would be very difficult to organize the structure and contents of the data in the data warehouse.

The essence of the data warehouse architecture: In addition to the benefit of visualization, the data model plays the role of a guideline, or plan, to implement the data warehouse. Traditionally, ER modeling has primarily focused on eliminating data redundancy and keeping consistency among the different data sources and applications. Consolidating the data models of each business area before the real implementation can help assure that the result will be an effective data warehouse and can help reduce the cost of implementation.

Different approaches of data modeling: ER and dimensional modeling, although related, are very different from each other. There is much debate as to which method is best and the conditions under which a particular technique should be selected. There can be no definite answer on which is best, but there are guidelines on which would be the better selection in a particular set of circumstances or in a particular environment. In the following sections, we review and define the modeling techniques and provide some selection guidelines.

6.2 Data Modeling Techniques

Two data modeling techniques that are relevant in a data warehousing environment are ER modeling and dimensional modeling.

ER modeling produces a data model of the specific area of interest, using two basic concepts: *entities* and the *relationships* between those entities. Detailed ER models also contain *attributes*, which can be properties of either the entities or the relationships. The ER model is an abstraction tool because it can be used to understand and simplify the ambiguous data relationships in the business world and complex systems environments.

Dimensional modeling uses three basic concepts: *measures*, *facts*, and *dimensions*. Dimensional modeling is powerful in representing the requirements of the business user in the context of database tables.

Both ER and dimensional modeling can be used to create an abstract model of a specific subject. However, each has its own limited set of modeling concepts and associated notation conventions. Consequently, the techniques look different, and they are indeed different in terms of semantic representation. The following sections describe the modeling concepts and notation conventions for both ER modeling and dimensional modeling that will be used throughout this book.

6.3 ER Modeling

A prerequisite for reading this book is a basic knowledge of ER modeling. Therefore we do not focus on that traditional technique. We simply define the necessary terms to form some consensus and present notation conventions used in the rest of this book.

6.3.1 Basic Concepts

An ER model is represented by an ER diagram, which uses three basic graphic symbols to conceptualize the data: entity, relationship, and attribute.

6.3.1.1 Entity

An entity is defined to be a person, place, thing, or event of interest to the business or the organization. An entity represents a class of objects, which are things in the real world that can be observed and classified by their properties and characteristics. In some books on IE, the term *entity type* is used to represent classes of objects and *entity* for an instance of an entity type. In this book, we will use them interchangeably.

Even though it can differ across the modeling phases, usually an entity has its own business definition and a clear boundary definition that is required to describe what is included and what is not. In a practical modeling project, the project members share a definition template for integration and a consistent entity definition in a model. In high-level business modeling an entity can be very generic, but an entity must be quite specific in the detailed logical modeling.

Figure 12 on page 38 shows an example of entities in an ER diagram. A rectangle represents an entity and, in this book, the entity name is notated by capital letters. In Figure 12 on page 38 there are four entities: PRODUCT, PRODUCT MODEL, PRODUCT COMPONENT, and COMPONENT. The four diagonal lines on the corners of the PRODUCT COMPONENT entity represent the notation for an *associative* entity. An associative entity is usually to resolve the many-to-many relationship between two entities. PRODUCT MODEL and COMPONENT are independent of each other but have a business relationship between them. A product model consists of many components and a component is related to many product models. With just this business rule, we cannot tell which components make up a product model. To do that you can define a resolving entity. For example, consider PRODUCT COMPONENT in Figure 12 on page 38. The PRODUCT COMPONENT entity can provide the information about which components are related to which product model.

In ER modeling, naming entities is important for an easy and clear understanding and communications. Usually, the entity name is expressed grammatically in the form of a noun rather than a verb. The criteria for selecting an entity name is how well the name represents the characteristics and scope of the entity.

In the detailed ER model, defining a unique identifier of an entity is the most critical task. These unique identifiers are called *candidate keys*. From them we can select the key that is most commonly used to identify the entity. It is called the *primary key*.

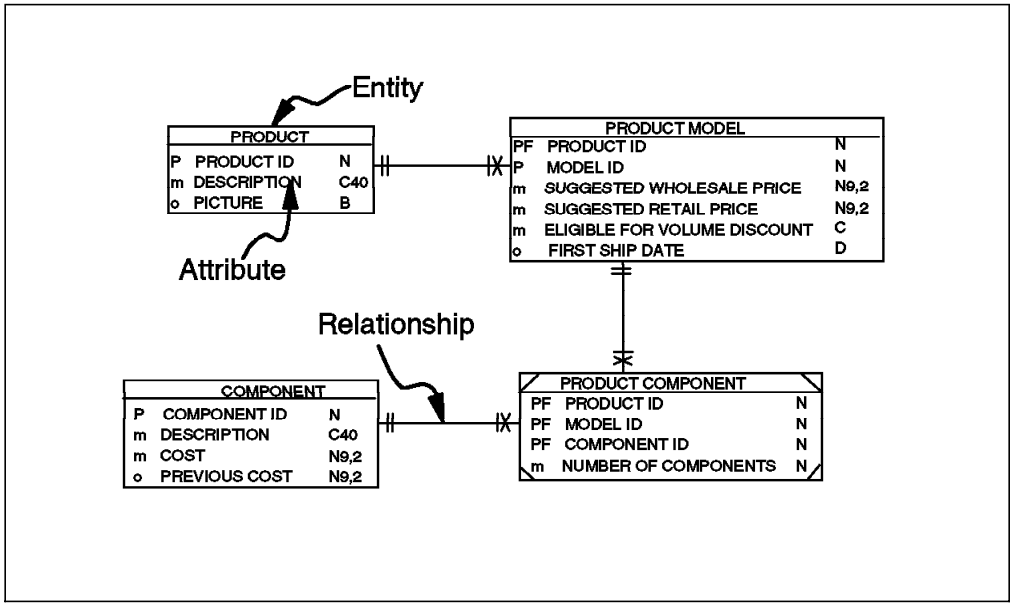


Figure 12. A Sample ER Model. Entity, relationship, and attributes in an ER diagram.

6.3.1.2 Relationship

A relationship is represented with lines drawn between entities. It depicts the structural interaction and association among the entities in a model. A relationship is designated grammatically by a verb, such as *owns, belongs*, and *has*. The relationship between two entities can be defined in terms of the cardinality. This is the maximum number of instances of one entity that are related to a single instance in another table and vice versa. The possible cardinalities are: one-to-one (1:1), one-to-many (1:M), and many-to-many (M:M). In a detailed (normalized) ER model, any M:M relationship is not shown because it is resolved to an associative entity.

Figure 12 shows examples of relationships. A high-level ER diagram has relationship names, but in a detailed ER diagram, the developers usually do not define the relationship name. In Figure 12, the line between COMPONENT and PRODUCT COMPONENT is a relationship. The notation (cross lines and short lines) on the relationship represents the cardinality.

When a relationship of an entity is related to itself, we can say that the relationship is *recursive*. Recursive relationships are usually developed either into associative entities or an attribute that references the other instance of the same entity.

When the cardinality of an entity is one-to-many, very often the relationship represents the dependent relationship of an entity to the other entity. In that case, the primary key of the parent entity is inherited into the dependent entity as some part of the primary key.

6.3.1.3 Attributes

Attributes describe the characteristics of properties of the entities. In Figure 12, Product ID, Description, and Picture are attributes of the PRODUCT entity. For clarification, attribute naming conventions are very important. An attribute name should be unique in an entity and should be self-explanatory. For example, simply saying date1 or date2 is not allowed, we must clearly define each. As examples, they could be defined as the order date and delivery date.

When an instance has no value for an attribute, the minimum cardinality of the attribute is zero, which means either *nullable* or *optional*. In Figure 12, you can see the characters *P, m, o, and F*. They stand for *primary key, mandatory, optional, and foreign key*. The Picture attribute of the PRODUCT entity is optional, which means it is nullable. A foreign key of an entity is defined to be the primary key of another entity. The Product ID attribute of the PRODUCT MODEL entity is a foreign key because it is the primary key of the PRODUCT entity. Foreign keys are useful to determine relationships such as the referential integrity between entities.

In ER modeling, if the maximum cardinality of an attribute is more than 1, the modeler will try to normalize the entity and finally elevate the attribute to another entity. Therefore, normally the maximum cardinality of an attribute is 1.

6.3.1.4 Other Concepts

A concept that seems frustrating to users is *domain*. However, it is actually a very simple concept. A domain consists of all the possible acceptable values and categories that are allowed for an attribute. Simply, a domain is just the whole set of the real possible occurrences. The format or data type, such as integer, date, and character, provides a clear definition of domain. For the enumerative type of domain, the possible instances should be defined. The practical benefits of domain is that it is imperative for building the data dictionary or repository, and consequently for implementing the database. For example, suppose that we have a new attribute called *product type* in the PRODUCT entity and the number of product types is fixed and with a value of CellPhone and Pager. The product types attribute forms an enumerative domain with instances of CellPhone and Pager, and this information should be included in the data dictionary. The attribute *first shop date* of the PRODUCT MODEL entity can be any date within specific conditions. For this kind of restrictive domain, the instances cannot be fixed, and the range or conditions should be included in the data dictionary.

Another important concept in ER modeling is *normalization*. Normalization is a process for assigning attributes to entities in a way that reduces data redundancy, avoids data anomalies, provides a solid architecture for updating data, and reinforces the long-term integrity of the data model. The third normal form is usually adequate. A process for resolving the many-to-many relationships is an example of normalization.

6.3.2 Advanced Topics in ER Modeling

In addition to the basic ER modeling concepts, three others are important for this book:

- Supertype and subtype
- Constraint statement
- Derivation

6.3.2.1 Supertype and Subtype

Entities can have subtypes and supertypes. The relationship between a supertype entity and its subtype entity is an *Isa* relationship. An Isa relationship is used where one entity is a generalization of several more specialized entities. Figure 13 on page 41 shows an example of supertype and subtype. In the figure, SALES OUTLET is the supertype of RETAIL STORE and CORPORATE SALES OFFICE. And, RETAIL STORE and CORPORATE SALES OFFICE are subtypes of SALES OUTLET. The notation of supertype and subtype is

represented by a triangle on the relationship. This notation is used by the IBM DataAtlas product.

Each subtype entity inherits attributes from its supertype entity. In addition to that, each subtype entity has its own distinctive attributes. In the example, subentities have Region ID and Outlet ID as inherited attributes. And, the subentities have their own attributes such as *number of cash registers* and *floor space* of the RETAIL STORE subentity.

The practical benefits of supertyping and subtyping are that it makes a data model more directly expressive. In Figure 13 on page 41, by just looking at the ER diagram we can understand that sales outlets are composed of retail stores and corporate sales offices.

The other benefits of supertyping and subtyping are that it makes a data model more ready to support flexible database developments. To transform supertype and subtype entities into tables, we can think of several implementation choices. We can make only one table within which an attribute is the indicator and many attributes are nullable. Otherwise, we can have only subtype tables to which all attributes of supertype are inherited. Another choice is to make tables for each entity. Each choice has its considerations. Through supertyping and subtyping, a very flexible data model can be implemented. Subtyping also makes the relationship clear. For example, suppose that we have a SALESPERSON entity and only corporate sales offices can officially have a salesperson. Without subtyping of SALES OUTLET into CORPORATE SALES OFFICE and RETAIL STORE, there is no way to express the constraints explicitly using ER notations.

Sometimes inappropriate use of supertyping and/or subtyping in ER modeling can cause problems. For example, a person can be a salesperson for the CelDial company as well as a customer. We might define person as being a supertype of employee and customer. But, in the practical world, it is not true. If we want a very generic model, we would better design a contract or association entity between *person* and *company*, or just leave it as customer and salesperson entities.

6.3.2.2 Constraints

Some constraints can be represented by relationships in the model. Basic referential integrity rules can be identified by relationships and their cardinalities. However, the more specific constraints such as ″Only when the occurrences of the parent entity ACCOUNT are checking accounts, can the occurrence of the child entity CHECK ACCOUNT DETAILS exist″ are not represented on an ER diagram. Such constraints can be added explicitly in the model by adding a constraint statement. This is particularly useful when we will have to show the temporal constraints, which also cannot be captured by relationship. For example, some occurrences of an entity have to be deleted when an occurrence of the other related entity is updated to a specific status. To define the life cycle of an entity, we need a constraint statement. Showing these types of specific conditions on an ER diagram is difficult.

If you define the basics of a language for expressing constraint statements, it will be very useful for communications among developers. For example, you could make a template for constraint statement with these titles:

- Constraint name and type
- Related objects (entity, relationship, attribute)
- Definition and descriptions

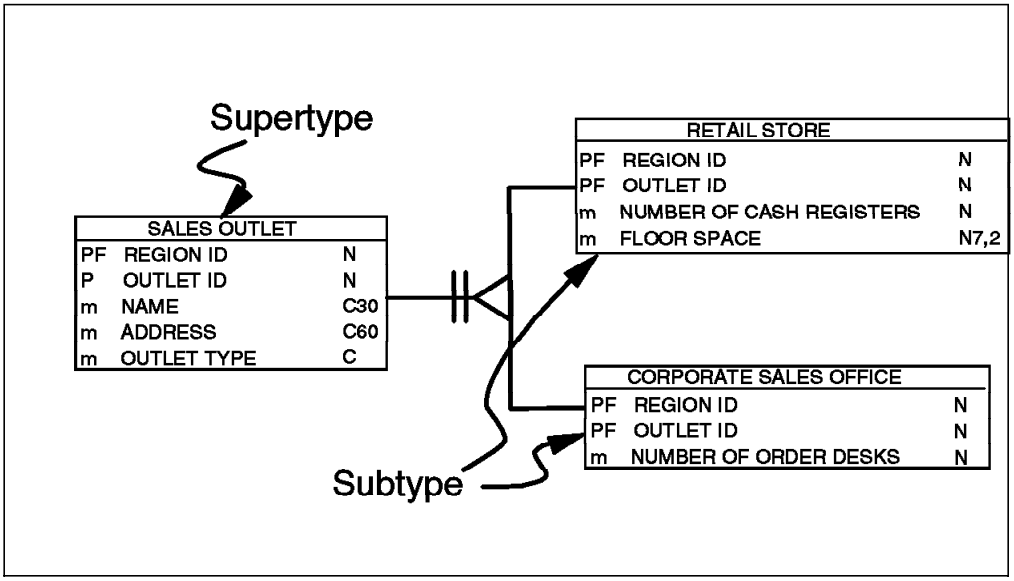


Figure 13. Supertype and Subtype

- Examples of the whole fixed number of instances

6.3.2.3 Derived Attributes and Derivation Functions

Derived attributes are less common in ER modeling for traditional OLTP applications, because they usually avoid having derived attributes at all. Data warehouse models, however, tend to include more derived attributes explicitly in the model. You can define a way to represent the derivation formula in the form of a statement. Through this form, you identify that an attribute is derived as well as providing explicitly the derivation function that is associated with the attribute.

As a matter of fact, all summarized attributes in the data warehouse are derived, because the data warehouse collects and consolidates data from source databases. As a consequence, the metadata should contain all of these derivation policies explicitly and users should have access to it.

For example, you can write a detailed derivation statement as follows:

- *Entity and attribute name* - SALES.Sales Volume.
- *Derivation source* - Sales Operational Database for each region. Related tables - SALES HISTORY, ...
- *Derivation function* - summarization of the gross sales of all sales outlets (formula: sales volume - returned volume - loss volume). Returned volume is counted only for the month.
- *Derivation frequency* - weekly (after closing of operational journaling on Saturday night)
- *Others*

Of course, you must not clutter up your ER model by explicitly presenting the derivation functions for all attributes. You need some compromise. Perhaps you can associate attributes derived from other attributes in the data warehouse with a derivation function that is explicitly added to the model. In any case, presenting the derivation functions is restricted to only where it helps to understand the model.

6.4 Dimensional Modeling

In some respects, dimensional modeling is simpler, more expressive, and easier to understand than ER modeling. But, dimensional modeling is a relatively new concept and not firmly defined yet in details, especially when compared to ER modeling techniques.

This section presents the terminology that we use in this book as we discuss dimensional modeling. For more detailed techniques, methodologies, and hints, refer to Chapter 8, “Data Warehouse Modeling Techniques” on page 81.

6.4.1 Basic Concepts

Dimensional modeling is a technique for conceptualizing and visualizing data models as a set of measures that are described by common aspects of the business. It is especially useful for summarizing and rearranging the data and presenting views of the data to support data analysis. Dimensional modeling focuses on numeric data, such as values, counts, weights, balances, and occurrences.

Dimensional modeling has several basic concepts:

- Facts
- Dimensions
- Measures (variables)

6.4.1.1 Fact

A fact is a collection of related data items, consisting of measures and context data. Each fact typically represents a business item, a business transaction, or an event that can be used in analyzing the business or business processes.

In a data warehouse, facts are implemented in the core tables in which all of the numeric data is stored.

6.4.1.2 Dimension

A *dimension* is a collection of members or units of the same type of views. In a diagram, a dimension is usually represented by an axis. In a dimensional model, every data point in the fact table is associated with one and only one member from each of the multiple dimensions. That is, dimensions determine the contextual background for the facts. Many analytical processes are used to quantify the impact of dimensions on the facts.

Dimensions are the parameters over which we want to perform Online Analytical Processing (OLAP). For example, in a database for analyzing all sales of products, common dimensions could be:

- Time
- Location/region
- Customers
- Salesperson
- Scenarios such as actual, budgeted, or estimated numbers

Dimensions can usually be mapped to nonnumeric, informative entities such as branch or employee.

Dimension Members: A dimension contains many dimension *members*. A dimension member is a distinct name or identifier used to determine a data item′s position. For example, all months, quarters, and years make up a time dimension, and all cities, regions, and countries make up a geography dimension.

Dimension Hierarchies: We can arrange the members of a dimension into one or more hierarchies. Each hierarchy can also have multiple hierarchy levels. Every member of a dimension does not locate on one hierarchy structure.

A good example to consider is the time dimension hierarchy as shown in Figure 14. The reason we define two hierarchies for time dimension is because a week can span two months, quarters, and higher levels. Therefore, weeks cannot be added up to equal a month, and so forth. If there is no practical benefit in analyzing the data on a weekly basis, you would not need to define another hierarchy for week.

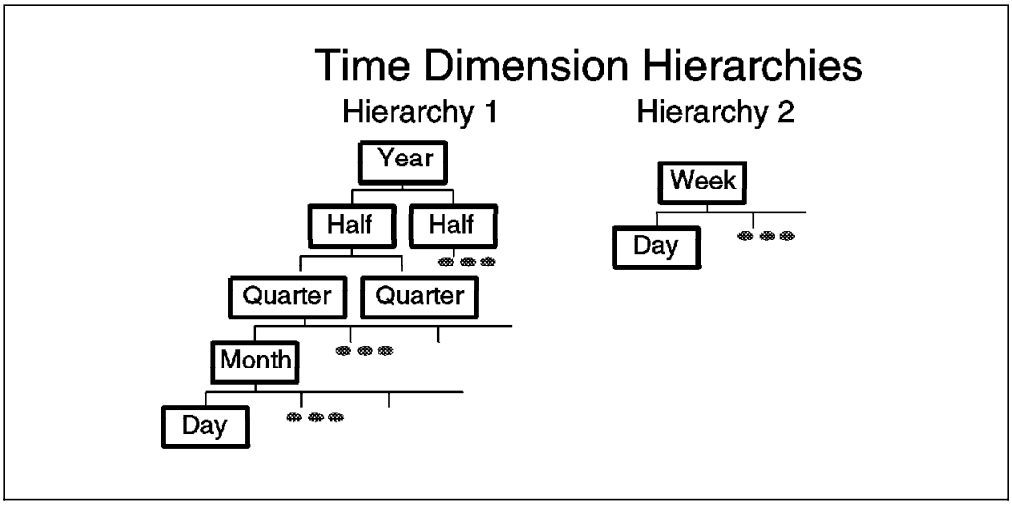


Figure 14. Multiple Hierarchies in a Time Dimension

6.4.1.3 Measure

A *measure* is a numeric attribute of a fact, representing the performance or behavior of the business relative to the dimensions. The actual numbers are called as *variables*. For example, measures are the sales in money, the sales volume, the quantity supplied, the supply cost, the transaction amount, and so forth. A measure is determined by combinations of the members of the dimensions and is located on facts.

6.4.2 Visualization of a Dimensional Model

The most popular way of visualizing a dimensional model is to draw a cube. We can represent a three-dimensional model using a cube. Usually a dimensional model consists of more than three dimensions and is referred to as a *hypercube*. However, a hypercube is difficult to visualize, so a cube is the more commonly used term.

In Figure 15 on page 44, the measurement is the volume of production, which is determined by the combination of three dimensions: location, product, and time. The location dimension and product dimension have their own two levels of hierarchy. For example, the location dimension has the region level and plant

level. In each dimension, there are members such as the east region and west region of the location dimension. Although not shown in the figure, the time dimension has its numbers, such as 1996 and 1997. Each subcube has its own numbers, which represent the volume of production as a measurement. For example, in a specific time period (not expressed in the figure), the Armonk plant in East region has produced 11,000 CellPhones, of model number 1001.

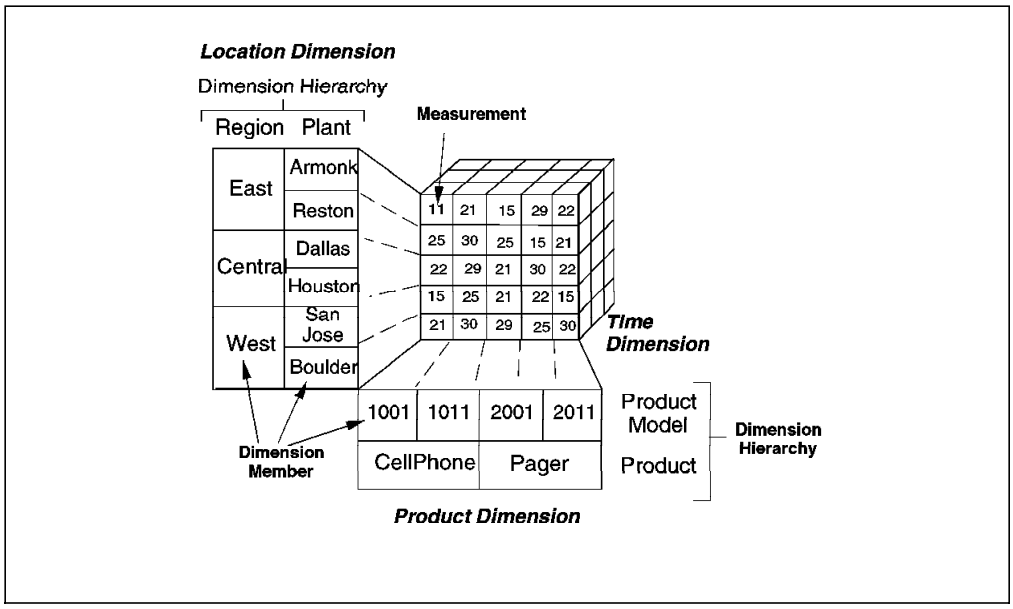


Figure 15. The Cube: A Metaphor for a Dimensional Model

6.4.3 Basic Operations for OLAP

Dimensional modeling is primarily to support OLAP and decision making. Let's review some of the basic concepts of OLAP to get a little better grasp of OLAP business requirements so that we can model the data warehouse more effectively.

Four types of operations are used in OLAP to analyze data. As we consider granularity, we can perform the operations of *drill down* and *roll up*. To browse along the dimensions, we use the operations *slice* and *dice*. Let's explore what those terms really mean.

6.4.3.1 Drill Down and Roll Up

Drill down and *roll up* are the operations for moving the view down and up along the dimensional hierarchy levels. With drill-down capability, users can navigate to higher levels of detail. With roll-up capability, users can zoom out to see a summarized level of data. The navigation path is determined by the hierarchies within dimensions. As an example, look at Figure 16 on page 45. While you analyze the monthly production report of the west region plants, you might like to review the recent trends by looking at past performance by quarter. You would be performing a roll-up operation by looking at the quarterly data. You may then wonder why the San Jose plant produced less than Boulder and would need more detailed information. You could then use the drill down-operation on the report by Team within a Plant to understand how the productivity of Team 2 (which is lower in all cases than the productivity for Team 1) can be improved.

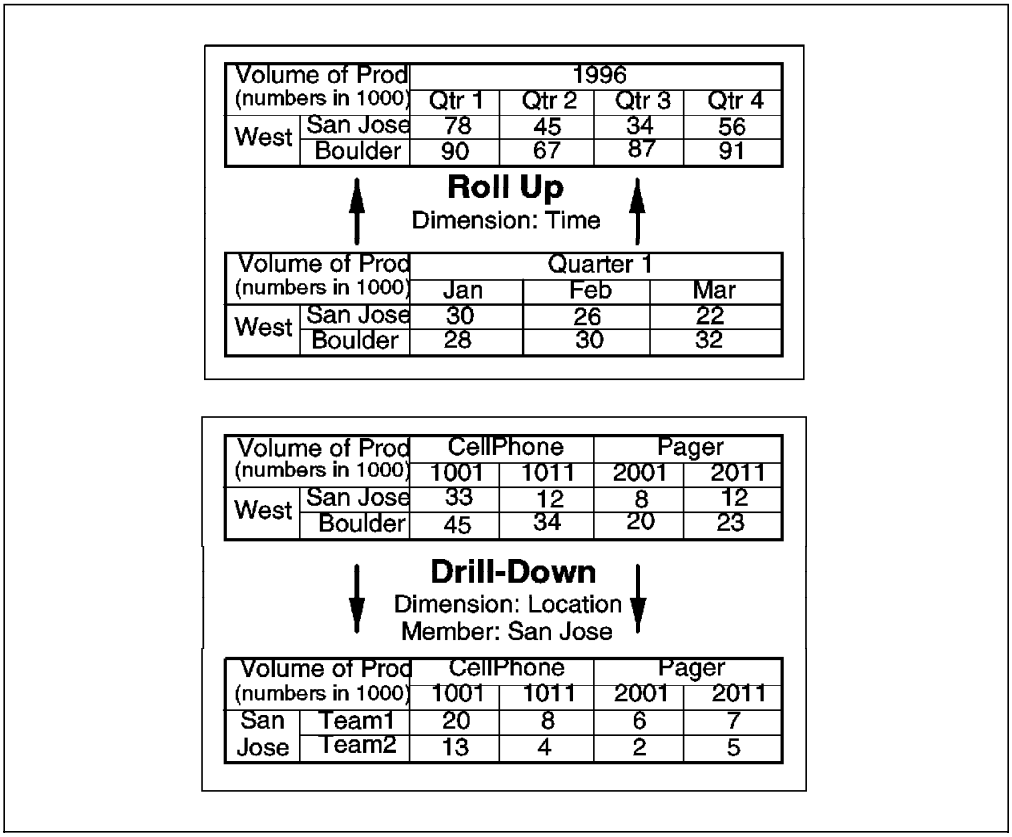


Figure 16. Example of Drill Down and Roll Up

6.4.3.2 Slice and Dice

Slice and *dice* are the operations for browsing the data through the visualized cube. Slicing cuts through the cube so that users can focus on some specific perspectives. Dicing rotates the cube to another perspective so that users can be more specific with the data analysis. Let's look at another example, using Figure 17 on page 46. You may be analyzing the production report of a specific month by plant and product, so you get the quarterly view of gross production by plant. You can then *change the dimension* from product to time, which is dicing. Now, you want to focus on the CellPhone only, rather than gross production. To do this, you can *cut off the cube* only for the CellPhone for the same dimensions, which is slicing.

Those are some of the key operations used in data analysis. To enable those types of operations requires that the data be stored in a specific way, and that is in a dimensional model.

6.4.4 Star and Snowflake Models

There are two basic models that can be used in dimensional modeling:

- Star model
- Snowflake model

Sometimes, the *constellation model* or *multistar model* is introduced as an extension of star and snowflake, but we will confine our discussion to the two basic structures. That is sufficient to explain the issues in dimensional modeling. This section presents only a basic introduction to the dimensional modeling techniques. For a detailed description, refer to Chapter 8, "Data Warehouse Modeling Techniques" on page 81.

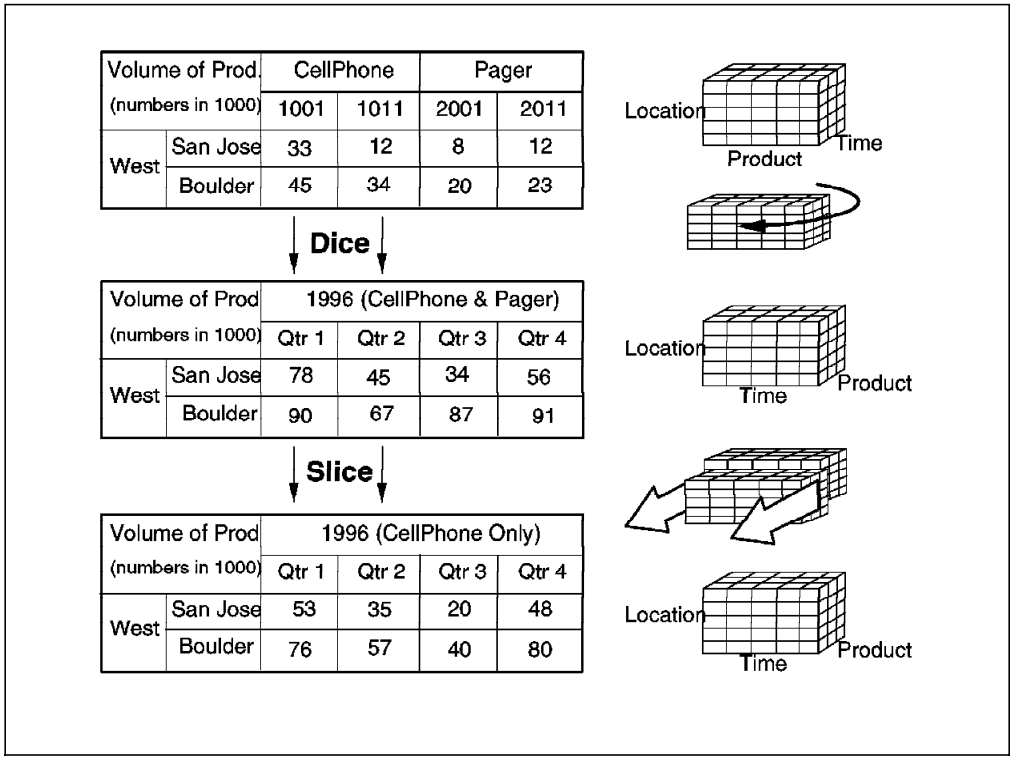


Figure 17. Example of Slice and Dice

6.4.4.1 Star Model

Star schema has become a common term used to connote a dimensional model. Database designers have long used the term star schema to describe dimensional models because the resulting structure looks like a star and the logical diagram looks like the physical schema. Business users feel uncomfortable with the term *schema*, so they have embraced the more simple sounding term of star model. In this book, we will also use the term *star model*.

The star model is the basic structure for a dimensional model. It typically has one large central table (called the *fact table*) and a set of smaller tables (called the *dimension tables*) arranged in a radial pattern around the fact table. Figure 18 on page 47 shows an example of a star schema. It depicts *sales* as a fact table in the center. Arranged around the fact table are the dimension tables of *time, customer, seller, manufacturing location*, and *product*.

Whereas the traditional ER model has an even and balanced style of entities and complex relationships among entities, the dimensional model is very asymmetric. Even though the fact table in the dimensional model is joined with all the other dimension tables, there is only a single join line connecting the fact table to the dimension tables.

6.4.4.2 Snowflake Model

Dimensional modeling typically begins by identifying facts and dimensions, after the business requirements have been gathered. The initial dimensional model is usually starlike in appearance, with one fact in the center and one level of several dimensions around it.

The snowflake model is the result of decomposing one or more of the dimensions, which sometimes have hierarchies themselves. We can define the many-to-one relationships among members within a dimension table as a

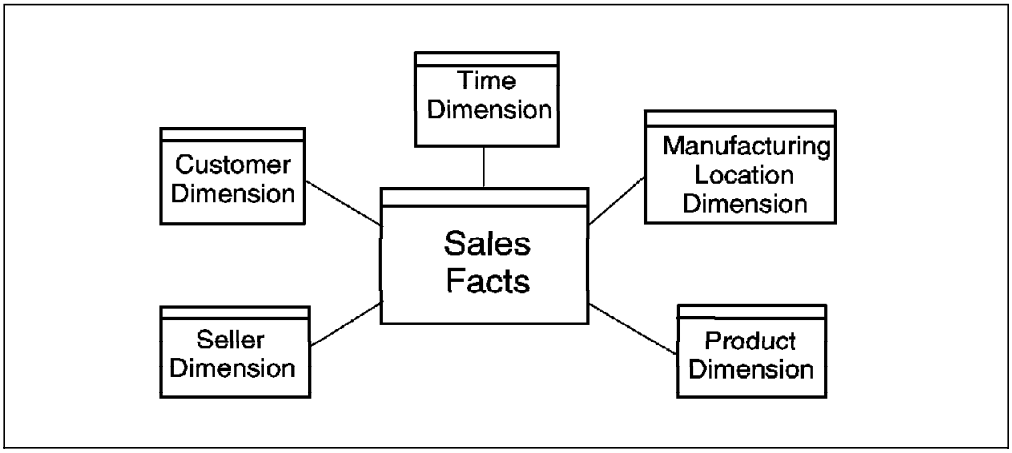


Figure 18. Star Model.

separate dimension table, forming a hierarchy. For example, the seller dimension in Figure 18 on page 47 is decomposed into subdimensions *outlet, region, and outlet type* in Figure 19 on page 48. This type of model is derived from the star schema and, as can be seen, looks like a snowflake.

The decomposed snowflake structure visualizes the hierarchical structure of dimensions very well. The snowflake model is easy for data modelers to understand and for database designers to use for the analysis of dimensions. However, the snowflake structure seems more complex and could tend to make the business users feel more uncomfortable working with it than with the simpler star model. Developers can also elect the snowflake because it typically saves data storage. Consider a banking application where there is a very large account table for one of the dimensions. You can easily expect to save quite a bit of space in a table of that size by not storing the very frequently repeated text fields, but rather putting them once in a subdimension table. Although the snowflake model does save space, it is generally not significant when compared to the fact table. Most database designers do not consider the savings in space to be a major decision criterion in the selection of a modeling technique.

6.4.5 Data Consolidation

Another major criterion for the use of OLAP is the fast response time for ad hoc queries. However, there could still be performance issues depending on the structure and volume of data. For a consistently fast response time, data *consolidation (precalculation* or *preaggregation)* is required. By precalculating and storing all subtotals before the query is issued, you can reduce the number of records to be retrieved for the query and maintain consistent and fast performance. The trade-off is that you will have to know how the users typically make their queries to understand how to consolidate. When users drill down to details, they typically move along the levels of a dimension hierarchy. Therefore, that provides the paths to consolidate or precalculate the data.

6.5 ER Modeling and Dimensional Modeling

The two techniques for data modeling in a data warehouse environment sometimes look very different from each other, but they have many similarities. Dimensional modeling can use the same notation, such as entity, relationship, attribute, and primary key. And, in general, you can say that a fact is just an entity in which the primary key is a combination of foreign keys, and the foreign

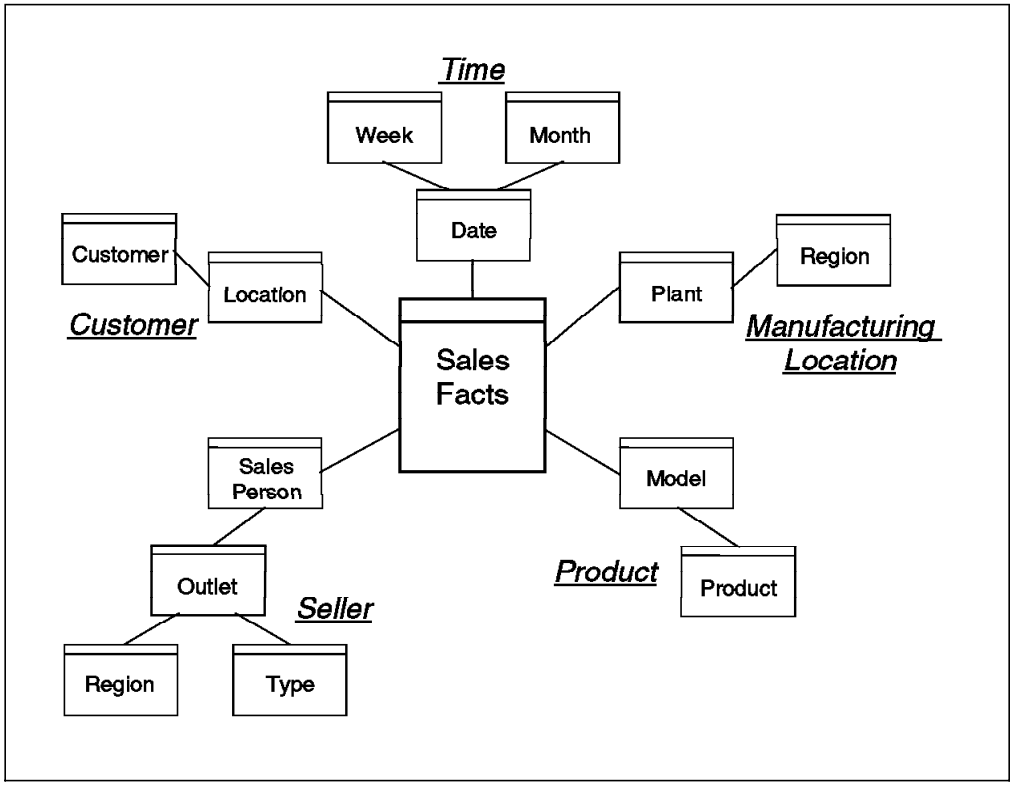


Figure 19. Snowflake Model

keys reference the dimensions. Therefore, we could say that dimensional modeling is a special form of ER modeling. An ER model provides the structure and content definition of the informational needs of the corporation, which is the base for designing the data warehouse.

This chapter defines the basic differences between the two primary data modeling techniques used in data warehousing. A conclusion that can be drawn from the discussion is that the two techniques have their own strengths and weaknesses, and either can be used in the appropriate situation.

Chapter 7. The Process of Data Warehousing

This chapter presents a basic methodology for developing a data warehouse. The ideas presented generally apply equally to a data warehouse or a data mart. Therefore, when we use the term *data warehouse* you can infer *data mart*. If something applies only to one or the other, that will be explicitly stated. We focus on the process of data modeling for the data warehouse and provide an extended section on the subject but discuss it in the larger context of data warehouse development.

The process of developing a data warehouse is similar in many respects to any other development project. Therefore, the process follows a similar path. What follows is a typical, and likely familiar, development cycle with emphasis on how the different components of the cycle affect your data warehouse modeling efforts. Figure 20 shows a typical data warehouse development cycle.

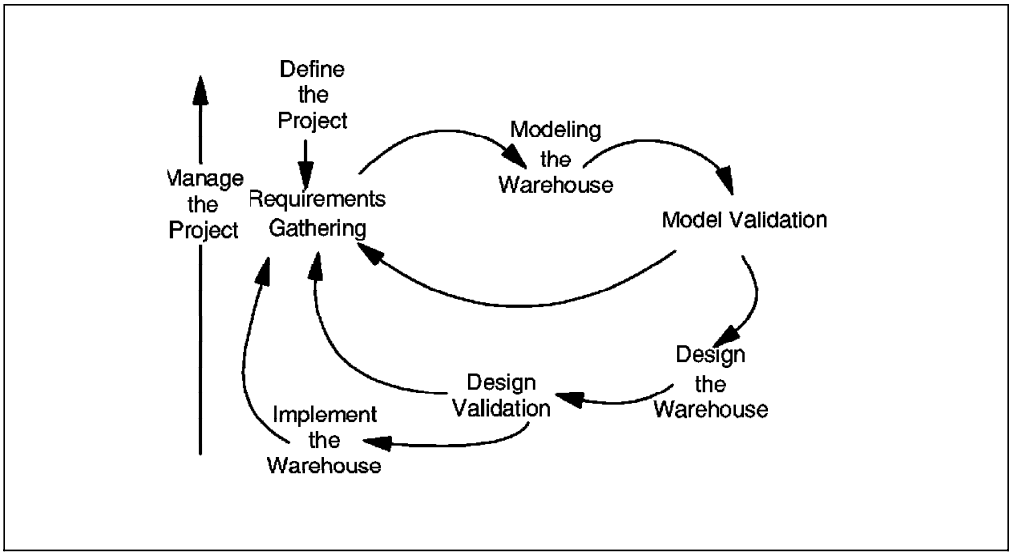


Figure 20. Data Warehouse Development Life Cycle

It is certainly true that there is no one correct or definitive life cycle for developing a data warehouse. We have chosen one simply because it seems to work well for us. Because our focus is really on modeling, the specific life cycle is not an issue here. What is essential is that we identify what you need to know to create an effective model for your data warehouse environment.

There are a number of considerations that must be taken into account as we discuss the data warehouse development life cycle. We need not dwell on them, but be aware of how they affect the development effort and understand how they will affect the overall data warehouse design and model.

- The life cycle diagram in Figure 20 seems to infer a single instance of a data warehouse. Clearly, this should be considered a logical view. That is, there could be multiple physical instances of a data warehouse involved in the environment. As an example, consider an implementation where there are multiple data marts. In this case you would iterate through the tasks in the life cycle for each data mart. This approach, however, brings with it an additional consideration, namely, the integration of the data marts. This integration can have an impact on the physical data, with considerations for

redundancy, inconsistency, and currency levels. Integration is also especially important because it can require integration of the data models for each of the data marts as well.

If dimensional modeling were being used, the integration might take place at the dimension level. Perhaps there could be a more global model that contains the dimensions for the organization. Then when data marts, or multiple instances of a data warehouse, are implemented, the dimensions used could be subsets of those in the global model. This would enable easier integration and consistency in the implementation.

- Data marts can be dependent or independent. In the previous consideration we addressed dependent data marts with their need for integration. Independent data marts are basically smaller in scope data warehouses that are stand-alone. In this case the data models can also be independent, but you must understand that this type of implementation can result in data redundancy, inconsistency, and currency levels.

The key message of the life cycle diagram is the iterative nature of data warehouse development. This, more than anything else, distinguishes the life cycle of a data warehouse project from other development projects. Whereas all projects have some degree of iteration, data warehouse projects take iteration to the extreme to enable fast delivery of portions of a warehouse. Thus portions of a data warehouse can be delivered while others are still being developed. In most cases, providing the user with some data warehouse function generates immediate benefits. Delivery of a data warehouse is not typically an all-or-nothing proposition.

Because the emphasis of this book is on modeling for the data warehouse, we have left out discussion about infrastructure acquisition. Although this would certainly be part of any typical data warehouse effort, it does not directly impact the modeling process.

Within each step of the process a number of techniques are identified for creating the model. As the focus here is on what to do more than how to do it, very little detail is given for these techniques. A separate chapter (see Chapter 8, “Data Warehouse Modeling Techniques” on page 81) is provided for those requiring detailed knowledge of the techniques outlined here.

7.1 Manage the Project

On the left side of the diagram in Figure 20 on page 49, you see a line entitled *Manage the Project*. As with any development project, there must be a management component, and this component exists from the beginning to the end of the project. The development of a data warehouse is no different in this respect. However, it is a project management component and not a data warehouse management component. The difference is that management of a project is finite in scope and is concerned with the building of the data warehouse, whereas management of a data warehouse is ongoing (just as management of any other aspect of your organization, such as inventory or facilities) and is concerned with the execution of the data warehousing processes.

7.2 Define the Project

In a typical project, high-level objectives are defined during the project definition phase. As well, limits are set on what will be delivered. This is commonly called the *scope* of the project.

In data warehouse development, although the project objectives need to be specific, the data warehouse requirements are typically defined in general statements. They should answer such questions as, ″What do I want to analyze, and why do I want to analyze it?″ By answering the why question, we get an understanding of the requirements that must be addressed and begin to gain insight into the users′ information requirements.

Data warehouse requirements contrast with typical application requirements, which will generally contain specific statements about which processes need to be automated. It is important that the requirements for data warehouse development not be too specific. If they are too specific, they may influence the way the data warehouse is designed to the point of excluding factors that seem irrelevant but may be key to the analysis being conducted.

One of the main reasons for defining the scope of a project is to prevent constant change throughout the life cycle as new requirements arise. In data warehousing, defining the scope requires special care. It is still true that you want to prevent your target from constantly changing as new requirements arise. However, two of the keys to a valuable data warehouse are its flexibility and its ability to handle the as yet unknown query. Therefore, it is essential that the scope be defined to recognize that the delivered data warehouse will likely be somewhat broader than indicated by the initial requirements. You are walking a tightrope between a scope that leads to an ever-changing target, incapable of being pinned down and declared complete, and one so rigid that it cannot adjust to the users′ ever-changing requirements.

7.3 Requirements Gathering

The traditional development cycle focuses on automating the process, making it faster and more efficient. The data warehouse development cycle focuses on facilitating the analysis that will change the process to make it more effective. Efficiency measures how much effort is required to meet a goal. Effectiveness measures how well a goal is being met against a set of expectations.

The requirements identified at this point in the development cycle are used to build the data warehouse model. But, the requirements of an organization change over time, and what is true one day is no longer valid the next. How then, do you know when you have successfully identified the user′s requirements? Although there is no definitive test, we propose that if your requirements address the following questions, you probably have enough information to begin modeling:

- Who (people, groups, organizations) is of interest to the user?
- What (functions) is the user trying to analyze?
- Why does the user need the data?
- When (for what point in time) does the data need to be recorded?
- Where (geographically, organizationally) do relevant processes occur?
- How do we measure the performance or state of the functions being analyzed?

There are many methods for deriving business requirements. In general, these methods can be placed in one of two categories: source-driven requirements gathering and user-driven requirements gathering (see Figure 21 on page 52).

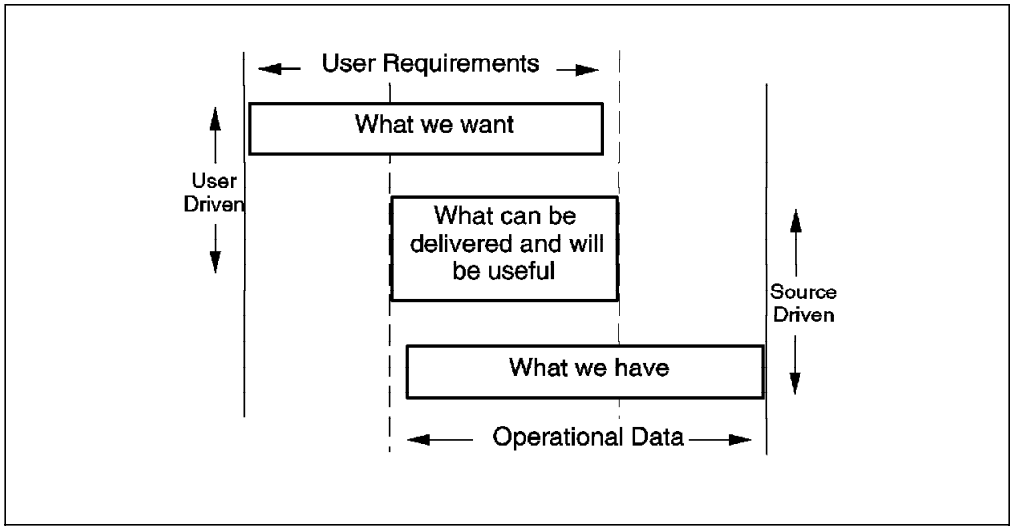


Figure 21. Two Approaches. Source-Driven and User-Driven Requirements Gathering

7.3.1 Source-Driven Requirements Gathering

Source-driven requirements gathering, as the name implies, is a method based on defining the requirements by using the source data in production operational systems. This is done by analyzing an ER model of source data if one is available or the actual physical record layouts and selecting data elements deemed to be of interest.

The major advantage of this approach is that you know from the beginning that you can supply all the data because you are already limiting yourself to what is available. A second benefit is that you can minimize the time required by the users in the early stages of the project.

Of course there are also disadvantages to this approach. By minimizing user involvement, you increase the risk of producing an incorrect set of requirements. Depending on the volume of source data you have, and the availability of ER models for it, this can also be a very time-consuming approach. Perhaps most important, some of the user's key requirements may need data that is currently unavailable. Without the opportunity to identify such requirements, there is no chance to investigate what is involved in obtaining external data. External data is data that exists outside the organization. Even so, external data can often be of significant value to the business users. Even though steps should be taken to ensure the quality of such data, there is no reason to arbitrarily exclude it from being used.

The result of the source-driven approach is to provide the user with what you have. We believe there are at least two cases where this is appropriate. First, relative to dimensional modeling, it can be used to drive out a fairly comprehensive list of the major dimensions of interest to the organization. If you ultimately plan to have an organizationwide data warehouse, this could minimize the proliferation of duplicate dimensions across separately developed data marts. Second, analyzing relationships in the source data can identify areas on which to focus your data warehouse development efforts.

7.3.2 User-Driven Requirements Gathering

User-driven requirements gathering is a method based on defining the requirements by investigating the functions the users perform. This is usually done through a series of meetings and/or interviews with users.

The major advantage to this approach is that the focus is on providing what is needed, rather than what is available. In general, this approach has a smaller scope than the source-driven approach. Therefore, it generally produces a useful data warehouse in a shorter timespan.

On the negative side, expectations must be closely managed. The users must clearly understand that it is possible that some of the data they need can simply not be made available. This is important because you do not want to limit what the user asks for. Outside-the-box thinking should be promoted when defining requirements for a data warehouse. This will prevent you from eliminating requirements simply because you think they might not be possible. If a user is too tightly focused, it is possible to miss useful data that is available in the production systems.

We believe user-driven requirements gathering is the approach of choice, especially when developing data marts. For a full-scale data warehouse, we believe it would be worthwhile to use the source-driven approach to break the project into manageable pieces, which may be defined as subject areas. The user-driven approach could then be used to gather the requirements for each subject area.

7.3.3 The CelDial Case Study

Throughout this chapter, we reference a case study (see Appendix A, “The CelDial Case Study” on page 163) to illustrate the steps in the process of creating a data warehouse model. In that case study, we create a set of corporatewide dimensions, using the source-driven requirements gathering approach. We then take the user-driven requirements gathering approach to define specific dimensional models. As each step in the process is presented, some component of the model is created. It would be well worthwhile to review that case study before continuing.

7.4 Modeling the Data Warehouse

Modeling the target warehouse data is the process of translating requirements into a picture along with the supporting metadata that represents those requirements. Although we separate the requirements and modeling discussions for readability purposes, in reality these steps often overlap. As soon as some initial requirements are documented, an initial model starts to take shape. As the requirements become more complete, so too does the model.

We must also point out that there is a distinction between completing the modeling phase and completing the model. At the end of the modeling phase, you have a complete picture of the requirements. However, only part of the metadata will have been documented. A model cannot truly be considered complete until the remainder of the metadata is identified and documented during the design phase.

For a discussion on selection of a modeling technique, refer to Chapter 8, “Data Warehouse Modeling Techniques” on page 81. The remainder of this section demonstrates the steps to follow in building a model of your data warehouse.

7.4.1 Creating an ER Model

We believe that ER modeling is generally well understood. In the circumstance that the physical data warehouse implementation is different enough from the dimensional model to warrant the creation of an ER model, standard ER modeling techniques apply.

Defining the dimensions for your organization is a worthwhile exercise. Creation of successive data marts will be easier if much of the dimension data already exists.

Let’s use the case study ER model (see Figure 92 on page 168) as an example. The first step is to remove all the entities that act as associative entities and all subtype entities. In the case study this includes *Product Component, Inventory, Order Line, Order, Retail Store, and Corporate Sales Office*. Be careful to create all the many-to-many relationships that replace these entities (see Figure 22).

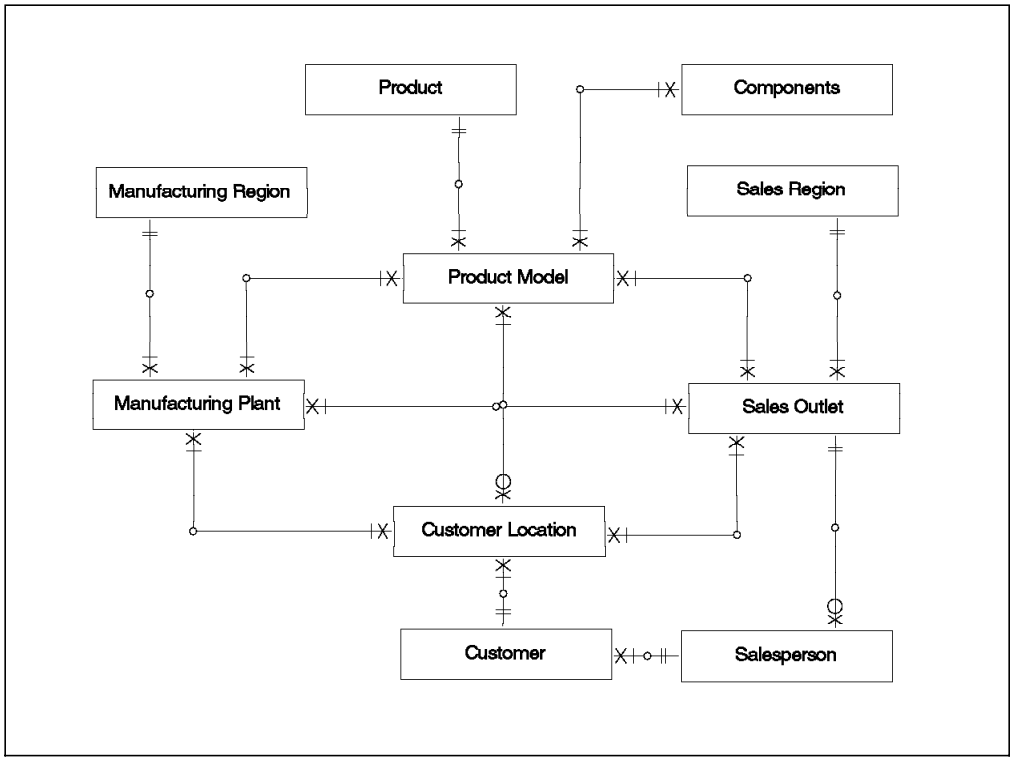


Figure 22. Corporate Dimensions: Step One. Removing subtypes and many-to-many relationships from an ER model.

The next step is to roll up the entities at the end of each of the many-to-many relationships into single entities. For each new entity, consider which attributes in the original entities would be useful constraints on the new dimension. Remember to consider attributes of any subtype entities removed in the first step. As well, because the model is a logical representation, we remove the individual keys and replace them with a generic key for each dimension (see Figure 23 on page 55). Physical keys will be assigned during the design phase.

In our case study example, note that rolling the salesperson up into the sales dimension implies (correctly) that the relationships among outlet, salesperson, and customer roll up into the sales to customer relationship. The many-to-many relationship between customer and sales prevents the erroneous rollup of customer into sales person and ultimately into sales.

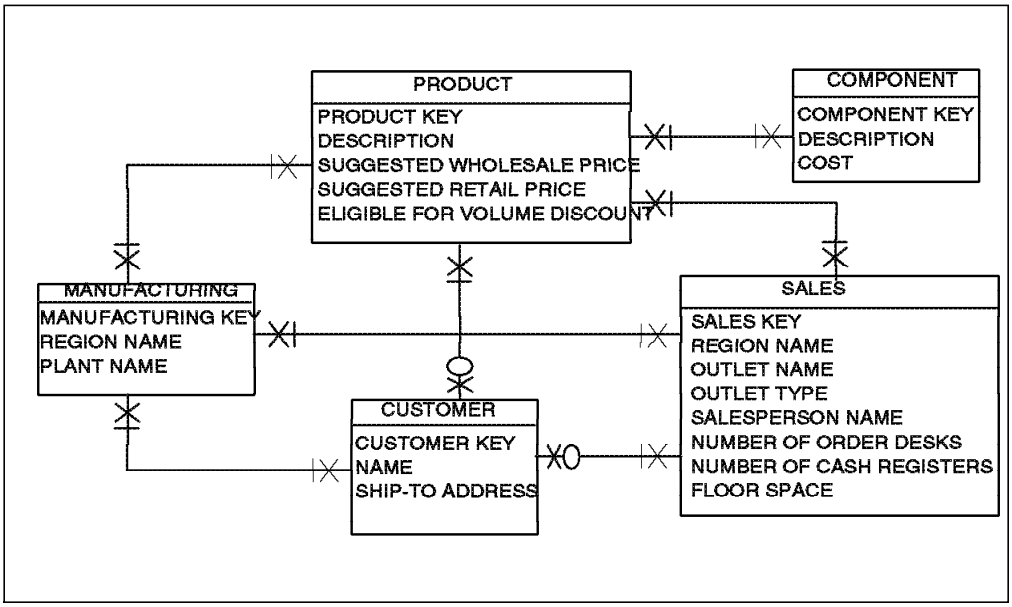


Figure 23. Corporate Dimensions: Step Two. Fully attributed dimensions for the organization.

7.4.2 Creating a Dimensional Model

The purpose of a data model is to represent a set of requirements for data in a clear and concise manner. In the case of a dimensional model, it is essential that the representation can be understood by the user. This model will be the basis for the analysis undertaken by a user and, if implemented properly, is how the user will see the data.

Although the structure should look like the model to the user, it may be physically implemented differently based on the technology used to create, maintain, and access it. We discuss this translation and completion of the model later in this chapter (see 7.5, “Design the Warehouse” on page 69).

The remainder of this section documents a set of steps to create a dimensional model that will be used to create the target data warehouse for the user′s data analysis requirements.

7.4.2.1 Dimensions and Measures

A user typically needs to evaluate, or analyze, some aspect of the organization′s business. The requirements that have been collected must represent the two key elements of this analysis: what is being analyzed, and the evaluation criteria for what is being analyzed. We refer to the evaluation criteria as measures and what is being analyzed as dimensions.

Our first step in creating a model is to identify the measures and dimensions within our requirements. A set of questions is defined in the case study that we

use as our sample requirements (see A.3.5, “What Do the Users Want?” on page 166). We restate these here:

1. What is the average quantity on-hand and reorder level this month for each model in each manufacturing plant?
2. What is the total cost and revenue for each model sold today, summarized by outlet, outlet type, region, and corporate sales levels?
3. What is the total cost and revenue for each model sold today, summarized by manufacturing plant and region?
4. What percentage of models are eligible for discounting and of those, what percentage is actually discounted when sold, by store, for all sales this week? This month?
5. For each model sold this month, what is the percentage sold retail, the percentage sold corporately through an order desk, and the percentage sold corporately by a salesperson?
6. Which models and products have not sold in the last week? In the last month?
7. What are the top five models sold last month by total revenue? By quantity sold? By total cost?
8. Which sales outlets had no sales recorded last month for each of the models in each of the three top five lists?
9. Which sales persons had no sales recorded last month for each of the models in each of the three top five lists?

By analyzing these questions, we define the dimensions and measures needed to meet the requirements (see Table 1).

Table 1. Dimensions, Measures, and Related Questions

Dimensions and Measures	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**
Dimensions									
Sales		X		X	X			X	X
Manufacturing	X		X						
Product	X	X	X		X	X	X	X	X
Measures									
Average quantity on hand	X								
Total cost		X	X				X		
Total revenue		X	X				X		
Quantity sold							X		
Percentage of models eligible for discount				X					
Percentage of models eligible for discount that are actually discounted				X					
Percentage of a model sold through a retail outlet					X				
Percentage of a model sold through a corporate sales office order desk					X				
Percentage of a model sold through a sales person					X				

Because we have already created the dimensions of CelDial (see Figure 23 on page 55), we do not go through the steps here to roll up the lower level entities

into each dimension. We only list the dimensions relevant to our requirements. If we did not have a corporate set of requirements to use here, we would have used the requirements generated from the questions in 7.4.2.1, “Dimensions and Measures” on page 55. This would have been a time-consuming exercise, but more importantly we would have had an incomplete set of dimensions and data. For example, we would have been unaware of the existence of the Customer and Component dimensions and the Number of Cash Registers and Floor Space attributes of the Sales dimension (see Figure 23 on page 55).

At this point we review the dimensions to ensure we have the data we need to answer our questions. No additional attributes are required for the sales and manufacturing dimensions. However, the product dimension as it stands cannot answer questions 2 and 3. To meet this need, we add the unit cost of a model to the product dimension. The derivation rule for this is defined in the case study (see A.3.4, “Defining Cost and Revenue” on page 165).

Based on the case study, there is interest in knowing the unit cost of a model at a point in time. We therefore conclude that a history of unit cost is necessary and add begin and end dates to fill out the product dimension (see Figure 24 on page 58).

7.4.2.2 Adding a Time Dimension

To properly evaluate any data it must be set in its proper context. This context always contains an element of time. Therefore we recommend the creation of a time dimension once for the organization. Be aware that adding time to another dimension as we did with product is a separate discussion. Here we only discuss time as a dimension of its own.

For most organizations, the lowest level of time that is relevant is an individual day. This is true for CelDial and so we choose day as our lowest level of granularity. Analyzing the requirements we can see a need for reporting by day, week, and month. Because we do not have more information about CelDial, we will not consider adding other attributes such as period, quarter, year, and day of week. When you initially create your time dimension, consider additional attributes such as those above and any others that may apply to your organization. We now have a time dimension that meets CelDial’s analysis requirements. This completes the dimensions we need to meet the documented case study requirements (see Figure 24 on page 58).

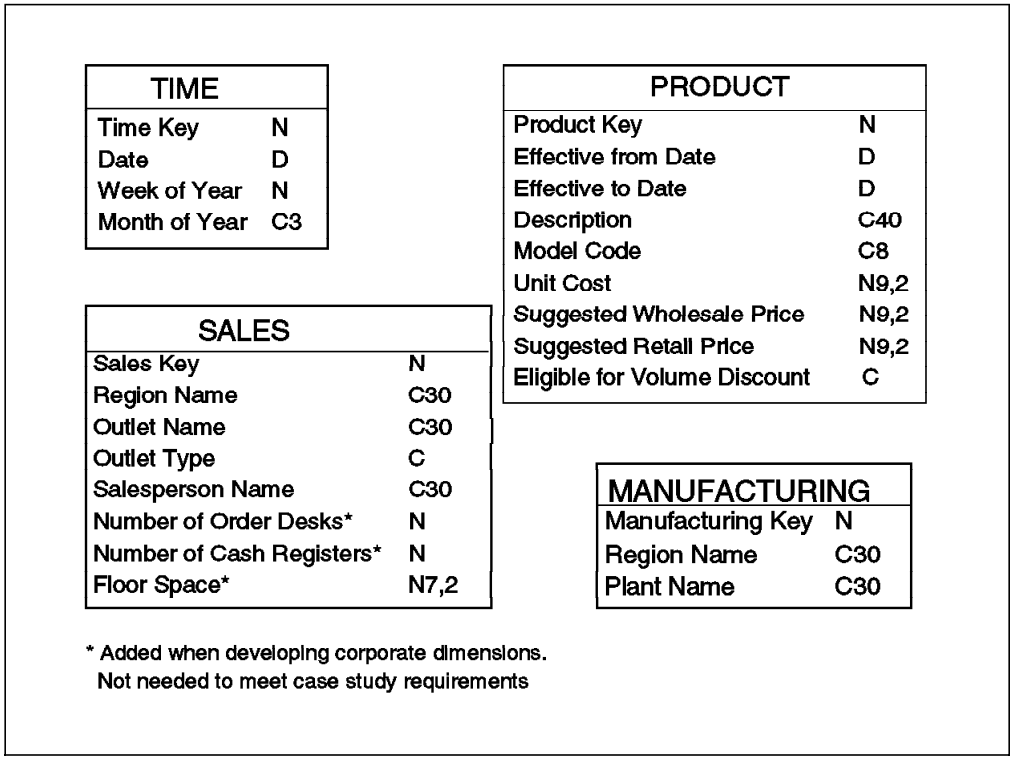


Figure 24. Dimensions of CelDial Required for the Case Study

7.4.2.3 Creating Facts

Together, one set of dimensions and its associated measures make up what we call a *fact*. Organizing the dimensions and measures into facts is the next step. This is the process of grouping dimensions and measures together in a manner that can address the specified requirements.

We will create an initial fact for each of the queries in the case study. For any measures that describe exactly the same set of dimensions, we will create only one fact (see Figure 25 on page 59).

Note that questions 6, 8, and 9 have no measures associated with them (see Table 1 on page 56). Had we not merged question 6 with questions 5 and 7 into fact 4, and questions 8 and 9 with question 2 into fact 2, these would produce facts containing no measures. Such facts are called *factless facts* because they only record that an event, in this case the sale of a product at a point in time (facts 2 and 3) at a specific location (fact 2 only), has occurred. No other measurement is required.

7.4.2.4 Granularity, Additivity, and Merging Facts

The granularity of a fact is the level of detail at which it is recorded. If data is to be analyzed effectively, it must all be at the same level of granularity. As a general rule, data should be kept at the highest (most detailed) level of granularity. This is because you cannot change data to a higher level than what you have decided to keep. You can, however, always roll up (summarize) the data to create a table with a lower level of granularity.

Closely related to the granularity issue is that of *additivity*, the ability of measures to be summarized. Measures fall into three categories: fully additive, nonadditive, and semiadditive. An example of a nonadditive measure is a

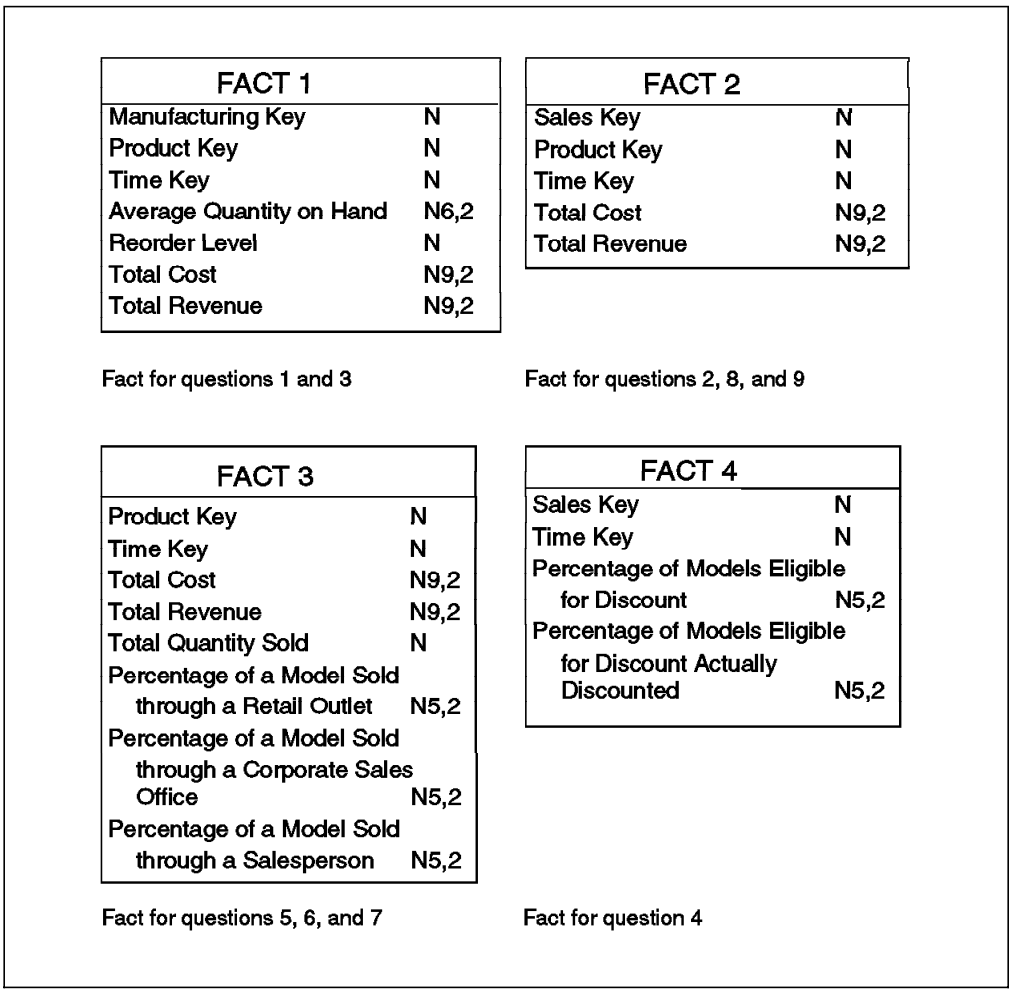


Figure 25. Initial Facts

percentage. You simply cannot add the percentages from two facts together and come up with a meaningful result. An example of a semiadditive measure is a balance. Although you can add the balances from two accounts to get a total balance, you cannot add two balances from the same account at two different points in time. Because the balance is additive only across some dimensions, we call it a semiadditive measure. A value that can be added across all dimensions is considered to be fully additive.

Additivity becomes important when you consider the possible summarizations that will occur on a fact table. Generally, it is desirable for measures to be fully additive. When they are not, you should consider breaking them down into their atomic elements.

Once you have assessed the granularity and additivity that exists in your facts, consider the possibility of merging facts. This may require changing the granularity of a particular fact. Usually, merging facts will expand the range of analyses that can be performed on the fact. This is because merging facts often implies adding dimensions to a fact.

Let's review the granularity and additivity of the facts we have generated and then consider the possibility of merging facts.

Granularity and Additivity: When reviewing fact 1, we immediately discover a problem with granularity. The average quantity on hand is a monthly figure, whereas the total cost and total revenue are daily figures. We must either split this into two facts or make the time dimension consistent. This means bringing time down to the lowest level of the two in question, which is day. However, because average quantity on hand is nonadditive, we have to store actual quantity on hand and let the query calculate an average. Making quantity on hand fully additive increases our range of analysis, so this seems to be the best choice.

Fact 2 also has a problem with the time dimension in that two different levels of granularity (day for query 2 and month for queries 8 and 9) exist. However, because all of the measures are fully additive, we will simply set the *grain* of time to a day. The grain is the fundamental atomic level of data to be represented in the fact table. A query can then handle any summarization to the monthly level.

Facts 3 and 4 both present difficult choices. As with the previous facts, we have two distinct grains of time. However, in this case neither grain can roll up into the other. To change the grain means setting time at the day level and summarizing up for both week and month. This becomes difficult because many of the measures are non additive. This seems to speak against changing the time granularity and in favor of splitting into multiple facts. The result, however, would be two facts containing exactly the same measures, with only the grain of the time dimension making a difference. This would certainly prove confusing when an analyst is trying to determine which fact to query against. Our preference would be to set the granularity at the day level and store the atomic elements of the percentage calculations for both of these facts. This is based on the idea that if the measures are the same, then having two facts is an unnecessary redundancy. Therefore in fact 3 we will replace the percentages with: quantity of model sold through a retail outlet, quantity of a model sold through a corporate sales office, and quantity of a model sold through a salesperson. Together with total quantity sold, which already exists in this fact, these measures will allow the percentages to be calculated. Similarly, in fact 4 we will replace the percentages with: number of models eligible for discount, quantity of models eligible for discount actually sold, and quantity of models sold at a discount.

Fact Consolidation: We have resolved the granularity and additivity problems with our facts (see Figure 26 on page 61). It is time to consider which, if any, can be consolidated. There are three main reasons for consolidating facts. First, it is easier for a user to find the data needed to satisfy a query if there are fewer places to look. Second, when you merge facts you expand the analysis potential because you can relate more measures to more dimensions at a higher level of granularity. Of course this is true only if it is valid to relate the dimensions and measures you are merging. Third, the fewer facts you have, the less administration there is to do.

The first step in evaluating the merge possibilities is to determine for each measure which additional dimensions can be added to increase its granularity.

Reviewing fact 1 we see that total cost and total revenue could be further broken down by the sales dimension. However, the same cannot be said for quantity on hand or reorder level. In fact, there is no finer breakdown for quantity on hand

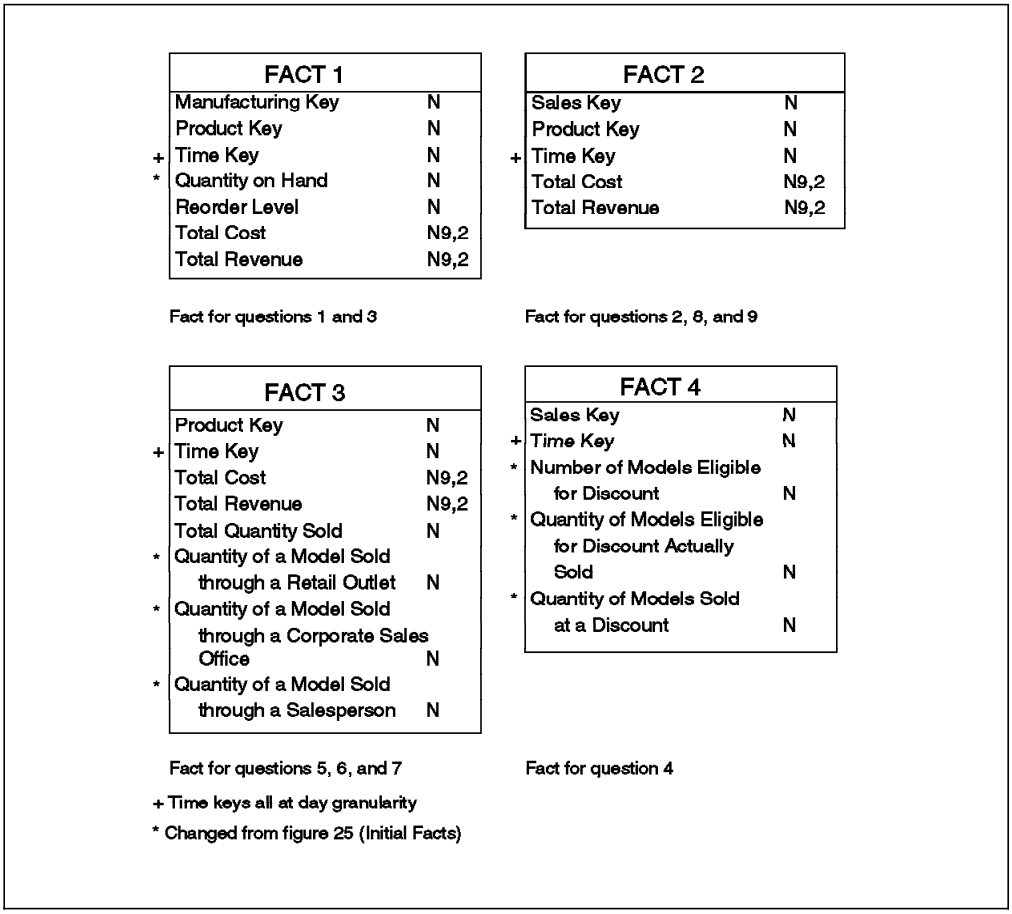


Figure 26. Intermediate Facts. Resolution of Granularity and Additivity Issues

than product and manufacturing. Therefore, we recommend no changes for fact 1.

Fact 2 already has all the dimensions present in facts 3 and 4. Therefore no further dimensions are necessary to allow a merger with facts 3 and 4. Let's examine facts 3 and 4 to see whether this merger is possible.

Adding the sales dimension to fact 3 necessitates some interesting changes. Certainly the total cost, total revenue, and total quantity sold can be further refined by adding the sales dimension. However, the sales dimension contains both outlet type and salesperson data. Using this structure we can classify the total quantity sold, negating the need to store the three individual totals. Facts 2 and 3 should definitely be merged (see Figure 27 on page 62).

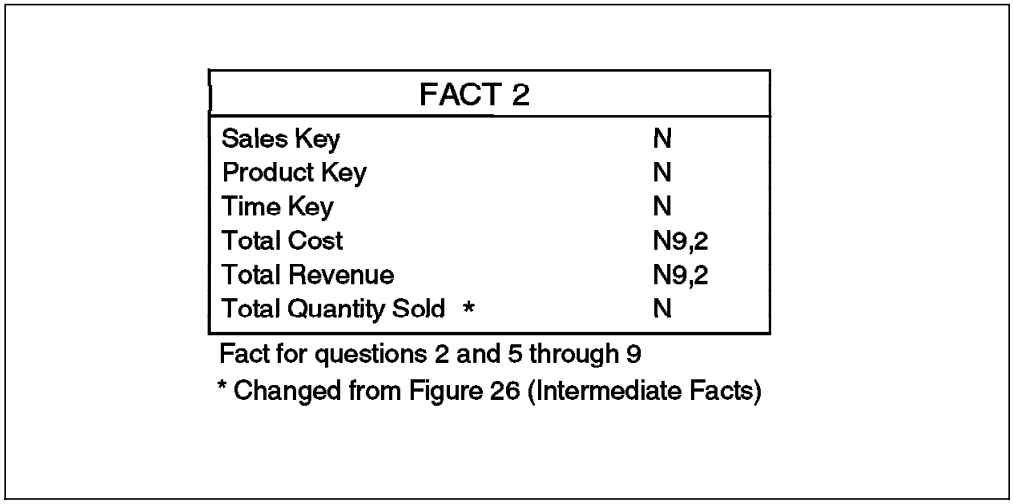


Figure 27. Merging Fact 3 into Fact 2

Adding the product dimension to fact 4 to facilitate the consolidation with fact 2 has similar results to our efforts with fact 3. The number of models eligible for discount can be calculated directly from the product dimension. Therefore it is no longer needed in the consolidated fact. Because the product dimension tells us whether an individual model is eligible for discount, we can use the total quantity sold (consolidated from fact 3) to represent the quantity of models eligible for discount actually sold. We have two options for quantity of models sold at a discount. One is to keep it as it is, which would meet the need. Another option is to record the discount amount and generate the quantity sold at a discount by adding up the quantity sold where the discount amount is not zero. Although this is not strictly required to meet the requirements, the potential for additional analysis makes this option attractive, and it is the one we recommend. The result is the consolidation of facts 2, 3, and 4 (see Figure 28).

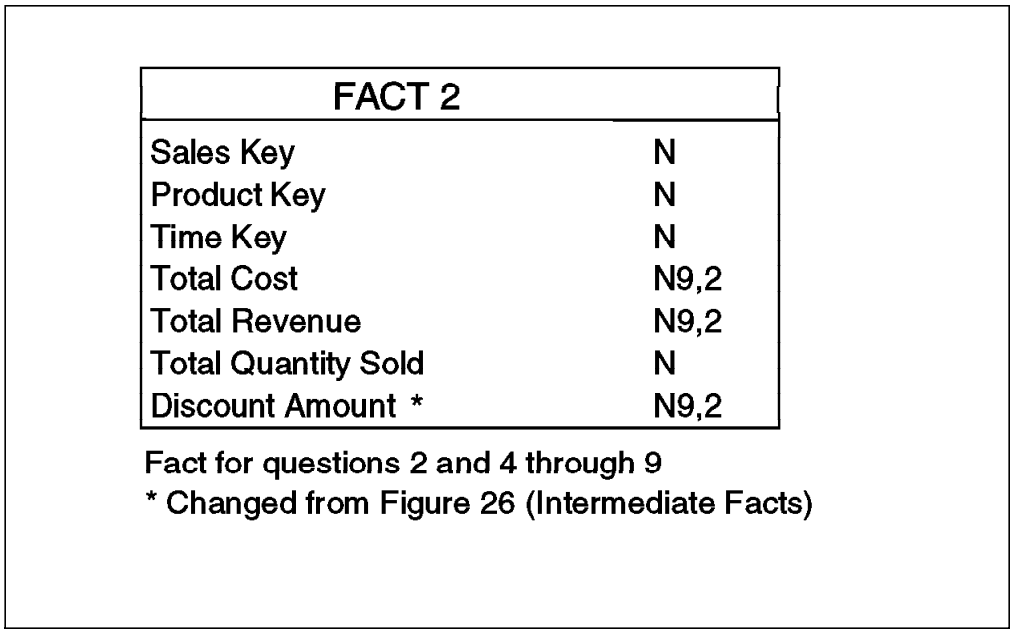


Figure 28. Merging Fact 4 into the Result of Fact 2 and Fact 3

One last step should be followed before declaring our model complete. The facts should be reviewed for opportunities to add other dimensions, thus increasing the potential for valuable analysis.

Again we see that fact 1 cannot be broken down any further. However, fact 2 still presents some opportunities. Both the manufacturing and customer dimensions can be applied to fact 2. The only other dimension we currently know about is the component dimension. Clearly it cannot be applied to fact 2. However, when we look at all the dimensions in fact 2, we see that there is one other possibility. The dimensions of fact 2 are: sales, product, manufacturing, customer, and time. All of these dimensions can be identified at the time an order is placed. The order would add one last level of granularity to fact 2. Although we have not, up to this point, considered order as a dimension, we do so now because of its effect on the granularity of fact 2. Because all of the measures in the fact are valid for an order, and because the increased granularity increases analysis potential, we recommend adding order to fact 2 as a dimension. Order is a dimension without attributes, so no dimension is actually created. We simply add order key to the fact. When we add a dimension in this manner, we refer to it as a *degenerate* dimension.

An interesting side effect of adding manufacturing to fact 2 is that we can now answer query 3 (from the requirements list in 7.4.2.1, “Dimensions and Measures” on page 55) through either fact. This gives us the option of removing total cost and total revenue from fact 1. For now, however, we will leave them there in case there is some value in relating inventory levels to sales activity.

Up to this point we have referred to our facts simply by number. Typically we name facts by what they represent. Since fact 1 represents inventory, we will call it the *inventory* fact. We will refer to fact 2 as the *sale* fact. Of course, this may be confusing as we already have a sales dimension. To clarify this, we will rename the sales dimension to the seller dimension. We now have our completely modeled facts (see Figure 29).

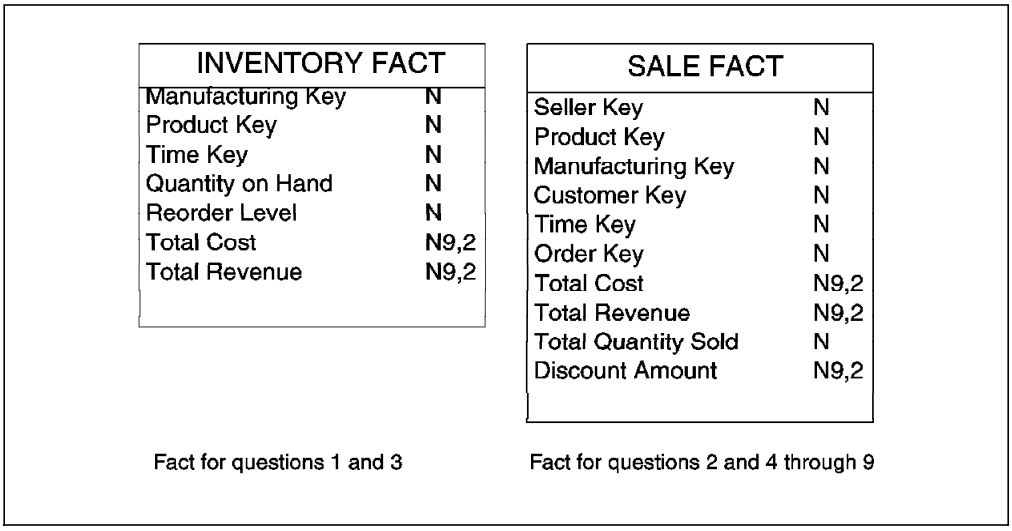


Figure 29. Final Facts

7.4.2.5 Integration with Existing Models

Once you have completed your facts and dimensions, follow the same process of setting granularity, additivity, and consolidation with data from your existing warehouse if you have one. The only significant difference is that you will be restricted in your ability to change already existing facts, dimensions, and measures. This is why it is so important to carry the process as far as possible up front.

In the case study we do not have an existing warehouse. Therefore, at this point we simply connect our dimensions to our facts to complete the pictures of our inventory model (see Figure 30) and our sales model (see Figure 31).

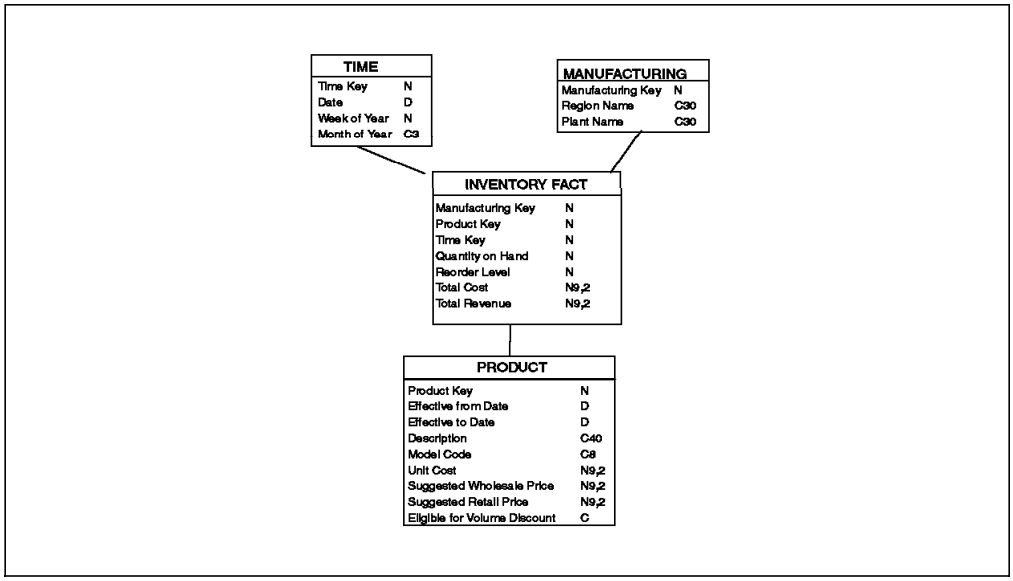


Figure 30. Inventory Model

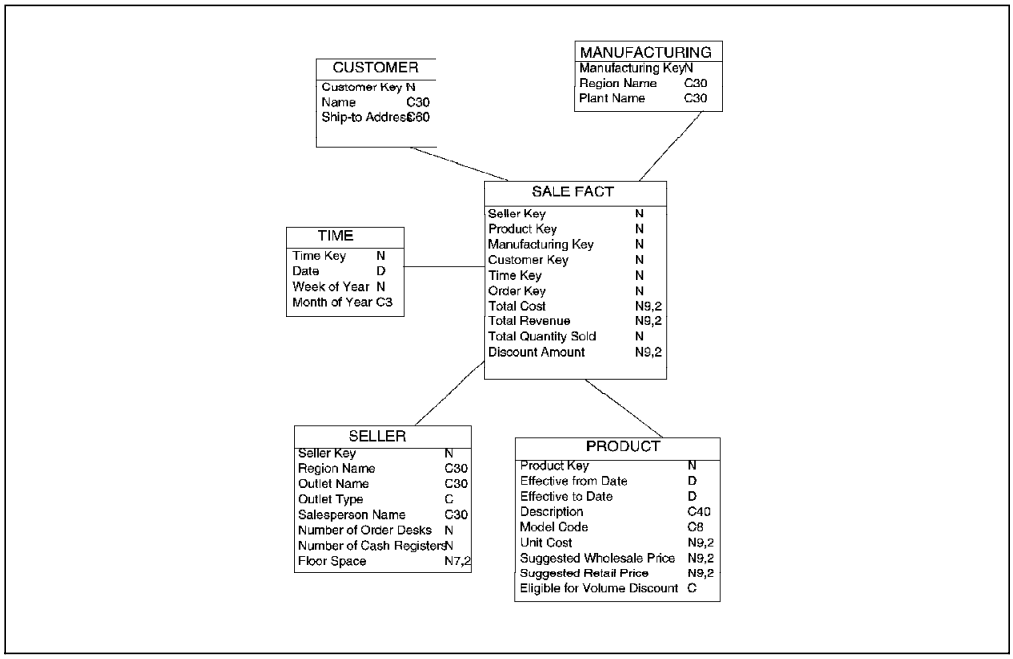


Figure 31. Sales Model

7.4.2.6 Sizing Your Model

Now that we have a model we can estimate how big our warehouse will be. At this point we can only give rough estimates of the size of our warehouse. We have enough information to estimate the size of the data in each of our tables. However, until some decisions are made regarding how these tables are physically implemented, items such as overhead and indices cannot be included.

To calculate the size of the data in a table, we simply multiply the number of rows times the length of each row. For our case study, we calculate row length by adding 4 bytes for each numeric or date attribute, the number of characters for a character attribute, and the number of digits in a decimal attribute divided by 2 and rounded up.

The method for determining the number of rows for each table varies. For seller, manufacturing, and customer, because we do not keep a history of changes, we simply use the total number of rows from the operational systems. For manufacturing there are seven plants. CelDial serves 3000 customers. There are three corporate sales offices, 15 retail stores, and 30 salespeople, for a total of 48 sellers.

For the remaining entities, the number of rows is dependent on how long we want to keep the data. In the case study, three complete years of data are required. Therefore no data can be deleted until the end of the fourth year. (If we needed three continuous years of data we could delete daily, weekly, or monthly all data that is more than three years old.)

There will be only one row per day for the time entity. Over four years, this will be 1,461 rows (4 years x 365 days + 1 day for the leap year).

There are 300 models of product. We must add to this a row for each of 10 changes per week. The result is 2,380 rows (300 models + 10 changes x 52 weeks x 4 years).

Our inventory fact will contain 3,068,100 rows (7 plants x 300 models x 1,461 days).

To determine the number of rows in our sales fact, we calculate corporate and retail sales separately because they have different volumes and days of operation. There will be 5,200,000 rows for corporate sales (500 sales x 10 models x 5 days x 52 weeks x 4 years). There will be 2,912,000 rows for retail sales (1000 sales x 2 models x 7 days x 52 weeks x 4 years). The total number of rows for the sales fact will be 8,112,000 (5,200,000 corporate sales rows + 2,912,000 retail sales rows). Multiplying the length of a row by the number of rows in each table and then adding these results together gives us a total of 431 MB (see Table 2 on page 66). Note that the size of the dimensions has no impact on the size of the warehouse when measured in megabytes. It is typical for the dimensions to be orders of magnitude smaller than facts. For this reason, sizing of the fact tables is often the only estimating done at this point. We only calculate the dimensions here for illustrative purposes.

Table 2. Size Estimates for CelDial's Warehouse				
Table	**Row Length**	**Number of Rows**	**Size (bytes)**	**Size**
Manufacturing	64	7	448	0.4 KB
Seller	107	48	5,136	5 KB
Time	15	1461	21,900	21.4 KB
Product	76	2,380	180,880	176.6 KB
Customer	94	3,000	282,000	275.4 KB
Inventory	30	3,068,100	92,043,000	89.9 MB
Sale	43	8,112,000	348,816,000	340.6 MB
Total	--	--	441,349,364	431 MB

This is a very preliminary estimate, to be sure. It does, however, enable technical staff to begin planning for its infrastructure requirements. This is important because, with the compressed life cycle of a warehouse, you will be needing that infrastructure soon.

7.4.3 Don't Forget the Metadata

In the traditional development cycle, a model sees only sparse use after completion, typically when changes need to be made, or when other projects require the data. In the warehouse, however, your model is used on a continuous basis. The users of the warehouse constantly reference the model to determine the data they want to use to analyze the organization. The rate of change of the data structure in a warehouse is much greater than that of operational data structures. Therefore, the technical users of the warehouse (administrators, modelers, designers, etc.) will also use your model on a regular basis.

This is where the metadata comes in. Far from just a pretty picture, the model must be a complete representation of the data you are storing, or it will be of little use to anybody.

At this point you cannot define all of the metadata. However, this does not mean you should wait until you can. To properly understand the model, and be able to confirm that it meets requirements, a user must have access to the metadata that describes the warehouse in business terms that are easily understood. Therefore, nontechnical metadata should be documented at this point. During the design phase, the technical metadata will be added to it.

At the warehouse level, a list should be provided of what is available in the warehouse. This list should contain the models, dimensions, facts, and measures available as these will all be used as initial entry points when a user begins analyzing data.

For each model, provide a name, definition, and purpose. The name simply gives the user something to focus on when searching. Usually, it is the same as the fact. The definition identifies what is modeled, and the purpose describes what the model is used for. The metadata for the model should also contain a list of dimensions, facts, and measures associated with it, as well as the name of a contact person so that users can get additional information when they have questions about the model.

A name, definition, and aliases must be provided for all dimensions, dimension attributes, facts, and measures. Aliases are necessary because it is often difficult to come to agreement on a common name for any widely used object. For dimensions and facts a contact person should be provided.

Metadata about a dimension should also include hierarchy, change rules, load frequency, and the attributes, facts, and measures associated with the dimension. The hierarchy defines the relationships between attributes of the dimension that identify the different levels that exist within it. For example, in the seller dimension we have the sales region, outlet type (corporate or retail), outlet, and salesperson as a hierarchy. This documents the roll-up structure of the dimension. Change rules identify how changes to attributes within a dimension are dealt with. In some instances, these rules can be different for individual attributes. Record change rules with the attributes when this is the case. The load frequency allows the user to understand whether data will be available when needed.

The attributes of a dimension are used to identify which facts the user wants to analyze. For example, in our case study, analyzing cost and revenue for the products that are very expensive to make might be done by only including facts where the unit cost attribute in the product dimension was greater than $500.00. For attributes to be used effectively, metadata about them should include the data type, domain, and derivation rules. At this point, a general indication of the data type (character, date, numeric, etc.) is sufficient. Exact data type definition can be done during design. The domain of an attribute defines the set of valid values that it can hold. For attributes that contain derived values, the rules for determining the value must be documented.

Metadata about a fact should include the load frequency, the measures and dimensions associated with the fact, and the grain of time for the fact. Although it is possible to derive the grain of time for a fact through its relationship to the time dimension, it is worthwhile explicitly stating it here. It is essential for proper analysis that this grain be understood.

Metadata about a measure should include its data type, domain, derivation rules, and the facts and dimensions associated with the measure.

To sum up the metadata required at this point, we provide a graphic representation of the metadata and the access paths users might travel when analyzing their data (see Figure 32 on page 68).

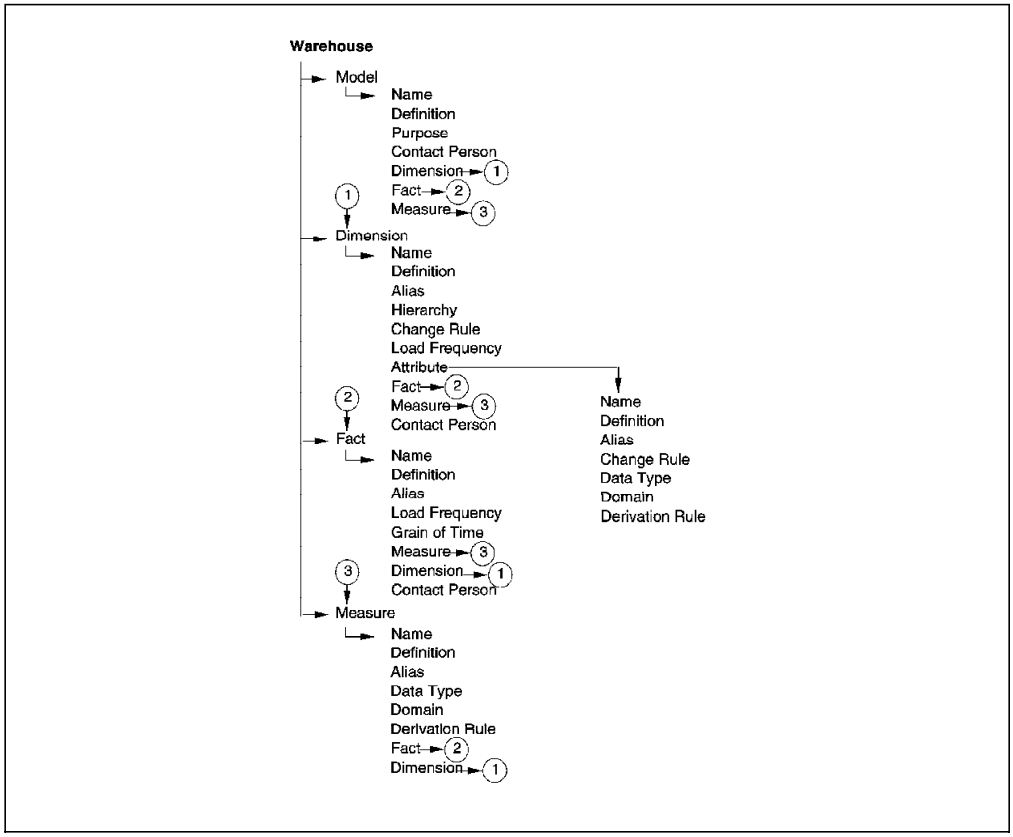


Figure 32. Warehouse Metadata. Metadata and user access paths at the end of the modeling phase

7.4.4 Validating the Model

Before investing a lot of time and effort in designing your warehouse, it is a good idea to validate your model with the user. The purpose of such a review is twofold. First, it serves to confirm that the model can actually meet the user's requirements. Second, and just as important, a review should confirm that the user can understand the model. Remember that once the warehouse is implemented, the user will be relying on the model on a regular basis to access data in the warehouse. No matter how well the model meets the user's requirements, your warehouse will fail if the user cannot understand the model and, consequently, cannot access the data.

Validation at this point is done at a high level. This model is reviewed with the user to confirm that it is understandable. Together with the user, test the model by resolving how you will answer some of the questions identified in the requirements.

It is almost certain that the model will not meet all of the user's requirements. This does not mean that you stop and go back to the beginning. Expect on your first cut of the model to meet perhaps 50% of the requirements. Take this 50% (or however much is validated) of the model and start working on the design. The remainder should be sent back to the requirements gathering stage. Either the requirements need to be better understood or, as is often the case, they have changed and need to be redefined. Usually, this will lead to additions, and possibly changes, to the model already created. In the mean time, the validated

portion of the model will go through the design phase and begin providing benefits to the user.

The iteration of development and the continued creation of partially complete models are the key elements that provide the ability to rapidly develop data warehouses.

7.5 Design the Warehouse

From the modeling perspective, our main focus in discussing the design phase of data warehouse development is identifying the additional metadata required to make the model complete. Aside from the metadata, there are a few cases where the design can impact the model. We identify these as well. We do not go beyond the design steps to discuss specific design techniques as this is beyond the scope of the book.

We begin this section with a short discussion on the differences in design for operational systems and data warehouse systems. This is followed with sections for each step in the design process. The focus of these sections is on the impact that the design techniques have on the model and its metadata. As the creation of a data mining application is primarily a design, not a modeling, function, we close this section with a discussion of data mining development.

7.5.1 Data Warehouse Design versus Operational Design

Once a model is created and validated, it is analyzed to determine the best way to physically implement it. Although similar in nature to modeling and design in the operational world, the actual steps in data warehousing are different. The discussion here assumes that operational models are typically ER models and the data warehousing models are dimensional models. You have probably already noticed that the data warehouse model looks more physical in nature than a model of an operational system. Probably the feature that most differentiates the data warehouse model from the logical operational model is the denormalization of the dimensions. Compare the product dimension in the dimensional model to the relevant entities in the ER model (see Figure 33 on page 70).

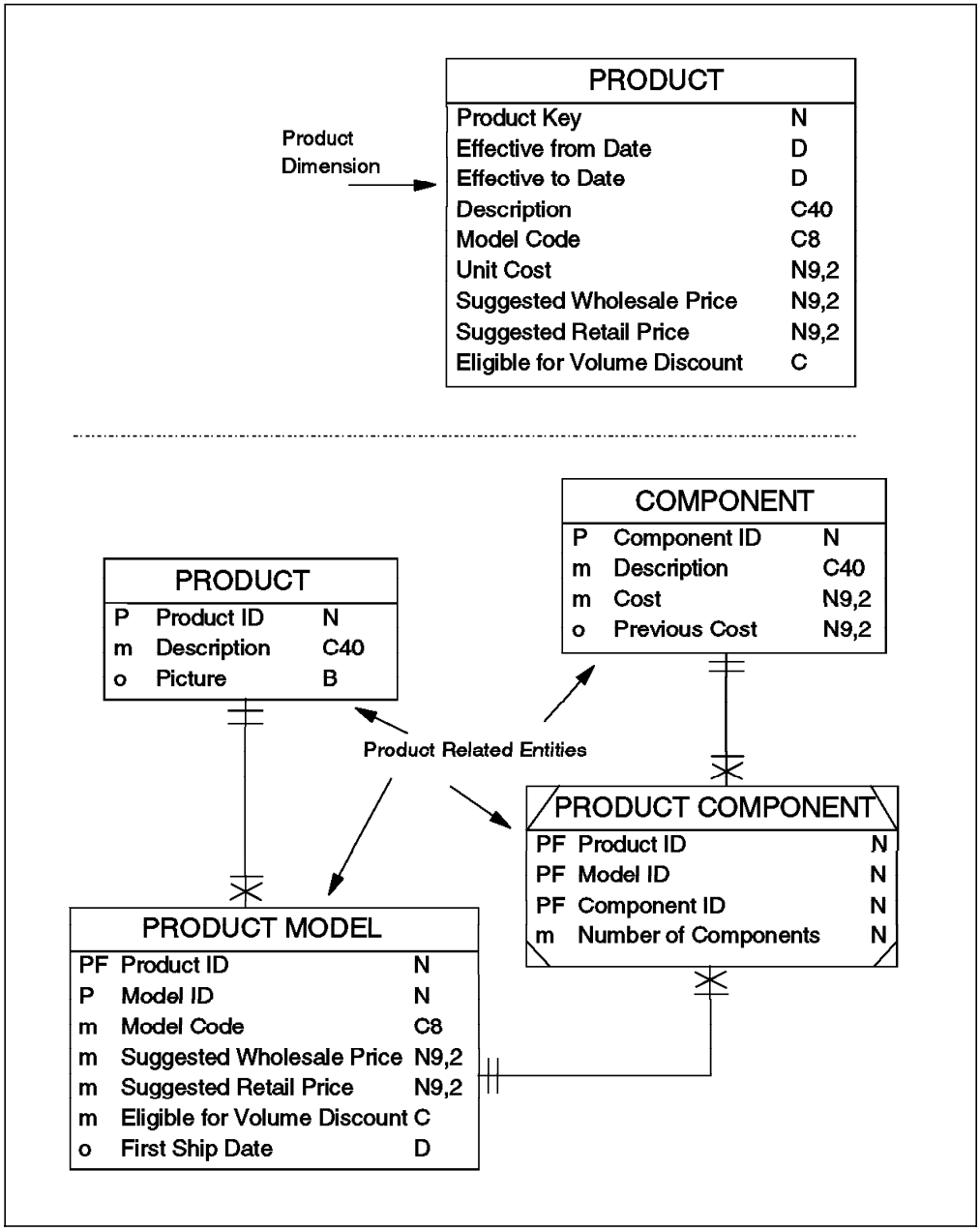


Figure 33. Dimensional and ER Views of Product-Related Data

The reason for this difference is the different role the model plays in the data warehouse. To the user, the data must look like the data warehouse model. In the operational world, a user does not generally use the model to access the data. The operational model is only used as a tool to capture requirements, not to access data.

Data warehouse design also has a different focus from operational design. Design in an operational system is concerned with creating a database that will perform well based on a well-defined set of access paths. Data warehouse design is concerned with creating a process that will retrieve and transform operational data into useful and timely warehouse data.

This is not to imply that there is no concern for performance in a data warehouse. On the contrary, due to the amount of data typically present in a

data warehouse, performance is an essential consideration. However, performance considerations cannot be handled in a data warehouse in the same way they are handled in operational systems. Access paths have already been built into the model due to the nature of dimensional modeling. The unpredictable nature of data warehouse queries limits how much further you can design for performance. After implementation, additional tuning may be possible based on monitoring usage patterns.

One area where design can impact performance is renormalizing, or snowflaking, dimensions. This decision should be made based on how the specific query tools you choose will access the dimensions. Some tools enable the user to view the contents of a dimension more efficiently if it is snowflaked while for other tools the opposite is true. As well, the choice to snowflake will also have a tool-dependent impact on the join techniques used to relate a set of dimensions to a fact. Regardless of the design decision made, the model should remain the same. From the user perspective, each dimension should have a single consolidated image.

7.5.2 Identifying the Sources

Once the validated portion of the model passes on to the design stage, the first step is to identify the sources of the data that will be used to load the model.

These sources should then be mapped to the target warehouse data model. Mapping should be done for each dimension, dimension attribute, fact, and measure. For dimensions and facts, only the source entities (for example, relational tables, flat files, IMS DBDs and segments) need be documented. For dimension attributes and measures, along with the source entities, the specific source attributes (such as columns and fields) must be documented.

Conversion and derivation algorithms must also be included in the metadata. At the dimension attribute and measure level, this includes data type conversion, algorithms for merging and splitting source attributes, calculations that must be performed, domain conversions, and source selection logic.

A domain conversion is the changing of the domain in the source system to a new set of values in the target. For example, in the operational system you may use codes for gender, such as 1=female and 2=male. You may want to convert this to female and male in the target system. Such a conversion should be documented in the metadata.

In some cases you may choose to load your target attribute from different source attributes based on certain conditions. Suppose you have a distributed sales organization and each location has its own customer file. However, your accounts receivable system is centralized. If you try to relate customer payments to sales data, you will likely have to pull some customer data from different locations based on where the customer does business. Source selection logic such as this must be included in the metadata.

At the fact and dimension level, conversion and derivation metadata includes the logic for merging and splitting rows of data in the source, the rules for joining multiple sources, and the logic followed to determine which of multiple sources will be used.

Identifying sources can also cause changes to your model. This will occur when you cannot find a valid source. Two possibilities exist. First, there simply is no

source that comes close to meeting the user's requirements. This should be very rare, but it is possible. If only a portion of the model is affected, remove that component and continue designing the remainder. Whatever portion of the model cannot be sourced must return to the requirements stage to redefine the need in a manner that can be met.

A more likely scenario is that there will be a source that comes close but is not exactly what the user had in mind. In the case study we have a product description but no model description. The model code is available to select individual models for analysis, but it is hardly user friendly. However, rather than not meet the requirement to perform analysis by model, model code will be used. If user knowledge of source systems is high, this may occur during the modeling stage, but often it occurs during design.

All of the metadata regarding data sources must be documented in the data warehouse model (see Figure 34 on page 77).

7.5.3 Cleaning the Data

Data cleaning has three basic components: validation of data, data enhancement, and error handling. Validation of data consists of a number of checks, including:

- Valid values for an attribute (domain check)
- Attribute valid in context of the rest of the row
- Attribute valid in context of related rows in this or other tables
- Relationship between rows in this and other tables valid (foreign key check)

This is not an exhaustive list. It is only meant to highlight the basic concepts of data validation.

Data enhancement is the process of cleaning valid data to make it more meaningful. The most common example is name and address information. Often we store name and address information for customers in multiple locations. Over time, these tend to become unsynchronized. Merging data for the customer is often difficult because the data we use to match the different images of the customer no longer matches. Data enhancement resynchronizes this data.

Error handling is a process that determines what to do with less than perfect data. Data may be rejected, stored for repair in a holding area, or passed on with its imperfections to the data warehouse. From a data model perspective, we only care about the data that is passed on to the data warehouse. The metadata for imperfect data should include statements about the data quality (types of errors) to be expected and the data accuracy (frequency of errors) of the data (see Figure 34 on page 77).

7.5.4 Transforming the Data

Data transformation is a critical step in any data warehouse development effort. Two major decisions must be made at this point: how to capture the source data, and a method for assigning keys to the target data. Along with these two decisions, you must generate a plan documenting the steps to get the data from source to target. From a modeling perspective, this is simply adding more metadata.

7.5.4.1 Capturing the Source Data

The first step in transformation is capturing the source data. Initially, a full copy of the data is required. Once this initial copy has been loaded, a means of maintaining it must be devised. There are four primary methods of capturing data:

- Full refresh
- Log capture
- Time-stamped source
- Change transaction files

A full refresh, as the name implies, is simply a full copy of the data to be moved into the target data warehouse. This copy may replace what is in the data warehouse, add a complete new copy at the new point in time, or be compared to the target data to produce a record of changes in the target.

The other three methods focus on capturing only what has changed in the source data. Log capture extracts relevant changes from the DBMS′s log files. If source data has been time stamped, the extract process can select only data that has changed since the previous extract was run. Some systems will produce a file of changes that have been made in the source. An extract can use this in the same manner it would use a log file.

From a modeling perspective, the method used should be documented in the metadata for the model. As well, the schedule of the extract should be documented at this point. Later, in the production environment, actual extract statistics will be added to this metadata (see Figure 34 on page 77).

7.5.4.2 Generating Keys

Key selection in the data warehouse is a difficult issue. It involves a trade-off between performance and management.

Key selection applies mainly to dimensions. The keys chosen for the dimensions must be the foreign keys of the fact.

There are two choices for dimension keys. Either an arbitrary key can be assigned, or identifiers from the operational system can be used. An arbitrary key is usually just a sequential number where the next available number is assigned when a new key is required.

To uniquely represent a dimension using identifiers from an operational system usually requires a composite key. A composite key is a key made up of multiple columns. An arbitrary key is one column and is almost always smaller than an operationally derived key. Therefore arbitrary keys will generally perform joins faster.

Generation of an arbitrary key is slightly more complex. If you get your key from the operational system, there is no need to determine the next available key. The exception to this is where history of a dimension is kept. In this case, when you use identifiers from an operational system, you must add an additional key because keys must be unique. One option is an arbitrary sequence number. Another is to add begin and end time stamps to the dimension key. Both of these options also work for an arbitrary key, but it is simpler just to generate a new arbitrary key when an entry in a dimension changes.

Once the history issue is considered, it certainly seems as if an arbitrary key is the way to go. However, the last factor in key selection is its impact on the fact table. When a fact is created, the key from each dimension must be assigned to it. If operationally derived keys, with time stamps for history, are used in the dimensions, there is no additional work when a fact is created. The linkage happens automatically. With arbitrary keys, or arbitrary history identifiers, a key must be assigned to a fact at the time the fact is created.

There are two ways to assign keys. One is to maintain a translation table of operational and data warehouse keys. The other is to store the operational keys and, if necessary, time stamps, as attribute data on the dimension.

The above discussion also applies to degenerate keys on the fact. The only difference is that there is no need to join on a degenerate key, thus diminishing the performance impact of an arbitrary key. The issue is more likely to come down to whether a user may need to know the value of a degenerate key for analysis purposes or that it is simply recorded to create the desired level of granularity.

The choice, then, is between better performance of an arbitrary key and easier maintenance of an operational key. The questions of how much better performance and how much more maintenance must be evaluated in your own organization.

Regardless of the choice you make, the keys, and the process that generates them, must be documented in the metadata (see Figure 34 on page 77). This data is necessary for the technical staff who administer and maintain the data warehouse. If the tools you use do not hide join processing, the user may need to understand this also. However, it is not recommended that a user be required to have this knowledge.

7.5.4.3 Getting from Source to Target

It is often the case that getting from source to target is a multiple step process. Rarely can it be completed in one step. Among the many reasons for creating a multiple step process to get from source to target are these:

- Sources to be merged are in different locations
- Not all data can be merged at once as some tables require outer joins
- Sources are stored on multiple incompatible technologies
- Complex summarization and derivation must take place

The point is simply that the process must be documented. The metadata for a model must include not only the steps of the process, but the contents of each step, as well as the reasons for it. It should look something like this:

1. Step 1 - Get Product Changes

   Objective of step
   : Create a table containing rows where product information has changed

   Inputs to step
   : Change transaction log for Products and Models, Product Component table, Component table, and the Product dimension table

   Transformations performed
   : For each change record, read the related product component and component rows. For each product model, the cost of each

component is multiplied by the number of components used to manufacture the model. The sum of results for all components that make up the model is the cost of that model. A key is generated for each record consisting of a sequential number starting with the next number after the highest used in the product dimension table. Write a record to the output table containing the generated key, the product and model keys, the current date, product description, model code, unit cost, suggested wholesale price, suggested retail price, and eligible for volume discount code.

Outputs of step

A work table containing new rows for the product dimension where there has been a change in a product or model

2. Step 2 - Get Component Changes

   Objective of step

   Create a table containing rows where component information has changed

   Inputs to step

   Change transaction log for Product Components and Components, Product table, Product Model table, the Product dimension table, and the work table from step 1

   Transformations performed

   For each change record, check that the product and model exist in the work table. If they do, the component change is already recorded so ignore the change record. If not, read the product and model tables for related information. For each product model, the cost of each component is multiplied by the number of components used to manufacture the model. The sum of results for all components that make up the model is the cost of that model. A key is generated for each record consisting of a sequential number starting with the next number after the highest used in the product dimension table. Add a record to the work table containing the generated key, the product and model keys, the current date, product description, model code, unit cost, suggested wholesale price, suggested retail price, and eligible for volume discount code.

   Outputs of step

   A work table containing additional new rows for the product dimension where there has been a change in the product component table or the component table

3. Step 3 - Update Product Dimension

   Objective of step

   Add changes to the Product dimension

   Inputs to step

   Work table from step 2

   Transformations performed

   For each row in the work table, a row is inserted into the product dimension. The effective to date is set to null. The effective to date of the previously current row is set to the day before the effective from date of the new row. A row is also written to a translation table containing the generated key, product key, model key, and change date.

Outputs of step

A translation table for use in assigning keys to facts and an updated product dimension

We do not suggest that this is the best (or even a good) transform method. The purpose here is to point out the type of metadata that should be recorded (see Figure 34 on page 77).

7.5.5 Designing Subsidiary Targets

Subsidiary targets are targets derived from the originally designed fact and dimension tables. The reason for developing such targets is performance. If, for example, a user frequently runs a query that sums across one dimension and scans the entire fact table, it is likely that a subsidiary target should be created with the dimension removed and measures summed to produce a table with less rows for this query.

Creating a subsidiary dimension should only be done if the original dimension will not join properly with a subsidiary fact. This is likely to be a tool-dependent decision.

Because this is a performance issue, rules should be defined for when a subsidiary target will be considered. Consider a maximum allowable time for a query before an aggregate is deemed necessary. You may also create a sliding scale of time it takes to run a query versus the frequency of the query.

Metadata for subsidiary targets should be the same as for the original facts and dimensions, with only the aggregates themselves being different. However, if your suite of tools can hide the subsidiary targets from the user and select them when appropriate based on the query, the metadata should be made visible only for technical purposes. The metadata should contain the reasons for creating the subsidiary target (see Figure 34 on page 77).

Often it is not possible to predict which subsidiary targets will be necessary at the design stage. These targets should not be created unless there is a clear justification. Rather than commit significant resources to them at this time, consider creating them as a result of monitoring efforts in the post-implementation environment.

Figure 34. The Complete Metadata Diagram for the Data Warehouse

7.5.6 Validating the Design

During the design stage you will create a test version of the production environment. When it comes time to validate the design with the user, hands-on testing is the best approach. Let the user try to answer questions through manipulation of the test target. Document any areas where the test target cannot provide the data requested.

Aside from testing, review with the user any additions and changes to the model that have resulted from the design phase to ensure they are understandable.

Similar to the model validation step, pass what works on to the implementation phase. What does not work should be returned to the requirements phase for clarification and reentry into modeling.

7.5.7 What About Data Mining?

Decisions in data warehouse modeling would typically not be affected by a decision to support data mining. However, the discussion on data mining, as one of the key data analysis techniques, is presented here for your information and completeness.

As stated previously, data mining is about creating hypotheses, not testing them. It is important to make this distinction. If you are really testing hypotheses, the

dimensional model will meet your requirements. It cannot, however, safely create a hypothesis. The reason for this is that by defining the dimensions of the data and organizing dimensions and measure into facts, you are building the hypotheses based on known rules and relationships. Once done, you have created a paradigm. To create a hypothesis, you must be able to work outside the paradigm, searching for patterns hidden in the unknown depths of the data.

There are, in general, four steps in the process of making data available for mining: data scoping, data selection, data cleaning, and data transformation. In some cases, a fifth step, data summarization, may be necessary.

7.5.7.1 Data Scoping

Even within the scope of your data warehouse project, when mining data you want to define a data scope, or possibly multiple data scopes. Because patterns are based on various forms of statistical analysis, you must define a scope in which a statistically significant pattern is likely to emerge. For example, buying patterns that show different products being purchased together may differ greatly in different geographical locations. To simply lump all of the data together may hide all of the patterns that exist in each location. Of course, by imposing such a scope you are defining some, though not all, of the business rules. It is therefore important that data scoping be done in concert with someone knowledgeable in both the business and in statistical analysis so that artificial patterns are not imposed and real patterns are not lost.

7.5.7.2 Data Selection

Data selection consists of identifying the source data that will be mined. Generally, the main focus will be on a transaction file. Once the transaction file is selected, related data may be added to your scope. The related data will consist of master files relevant to the transaction. In some cases, you will want to go beyond the directly related data and delve into other operational systems. For example, if you are doing sales analysis, you may want to include store staff scheduling data, to determine whether staffing levels, or even individual staff, create a pattern of sales of particular products, product combinations, or levels of sales. Clearly this data will not be part of your transaction, and it is quite likely the data is not stored in the same operational system.

7.5.7.3 Data Cleaning

Once you have scoped and selected the data to be mined, you must analyze it for quality. When cleaning data that will be mined, use extreme caution. The simple act of cleaning the data can remove or introduce patterns.

The first type of data cleaning (see 7.5.3, “Cleaning the Data” on page 72) is data validation. Validating the contents of a source field or column is very important when preparing data for mining. For example, if a gender code has valid values of M and F, all other values should be corrected. If this is not possible, you may want to document a margin of error for any patterns generated that relate to gender. You may also want to determine whether there are any patterns related to the bad data that can reveal an underlying cause.

Documenting relationships is the act of defining the relationships when adding in data such as the sales schedules in our data selection example. An algorithm must be developed to determine what part of the schedule gets recorded with a particular transaction. Although it seems clear that a sales transaction must be related to the schedule by the date and time of the sale, this may not be enough. What if some salespeople tend to start earlier than their shift and leave a little

earlier? As long as it all balances out, it may be easier for staff to leave the scheduling system alone, but your patterns could be distorted by such an unknown. Of course, you may not be able to correct the problem with this example. The point is simply that you must be able to document the relationship to be able to correctly transform the data for mining purposes.

The second type of data cleaning (see 7.5.3, “Cleaning the Data” on page 72), data enhancement, is risky when preparing data for mining. It is certainly important to be able to relate all images of a customer. However, the differences that exist in your data may also expose hidden patterns. You should proceed with enhancement cautiously.

7.5.7.4 Data Transformation

Depending on the capabilities of the tools you select to perform data mining, a set of relational tables or a large flat file may meet your requirements. Regardless, data transformation is the act of retrieving the data identified in the scoping and selection processes, creating the relationships and performing some of the validation documented in the cleaning process, and producing the file or tables to be mined. We say ″some of the validation″ because data that is truly incorrect should be fixed in the source operational system before transformation, unless you need to find patterns to indicate the cause of the errors. Such pattern searching should only be necessary, and indeed possible, if there is a high degree of error in the source data.

7.5.7.5 Data Summarization

There may be cases where you cannot relate the transaction data to other data at the granularity of the transaction; for example, the data needed to set the scope at the right level is not contained in the original transaction data. In such cases, you may consider summarizing data to allow the relationships to be built. However, be aware that altering your data in this way may remove the detail needed to produce the very patterns for which you are searching. You may want to consider mining at two levels when this summarization appears to be necessary.

7.6 The Dynamic Warehouse Model

In an operational system, shortly after implementation the system stabilizes and the model becomes static, until the next development initiative. But, the data warehouse is more dynamic, and it is possible for the model to change with no additional development initiative simply because of usage patterns.

Metadata is constantly added to the data warehouse from four sources (see Figure 35 on page 80). Monitoring of the warehouse provides usage statistics. The transform process adds metadata about what and how much data was loaded and when it was loaded. An archive process will record what data has been removed from the warehouse, when it was removed, and where it is stored. A purge process will remove data and update the metadata to reflect what remains in the data warehouse.

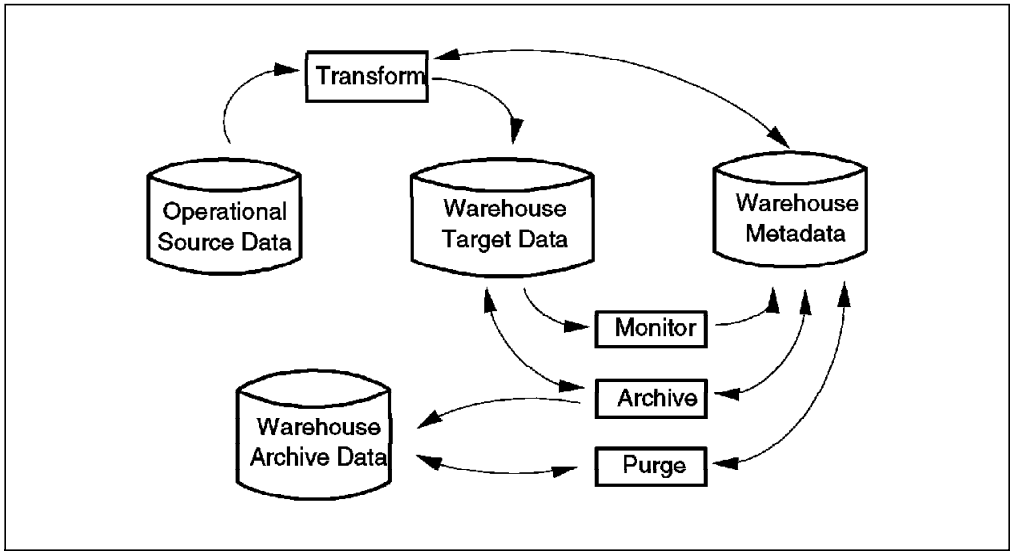


Figure 35. Metadata Changes in the Production Data Warehouse Environment

Based on usage and performance statistics or requests from users a data warehouse administrator may add or remove aggregates or alter archive and purge schedules. Such changes must also be reflected in the metadata (see Figure 34 on page 77).

Once a data warehouse is implemented, usage of it will spawn new requests and requirements. This will start another cycle of development, continuing the iterative and evolutionary process of building the data warehouse. As you can see, the data model is a living part of a data warehouse. Through the entire life cycle of the data warehouse, the data model is both maintained and used (see Figure 36). The process of data warehouse modeling can be truly endless.

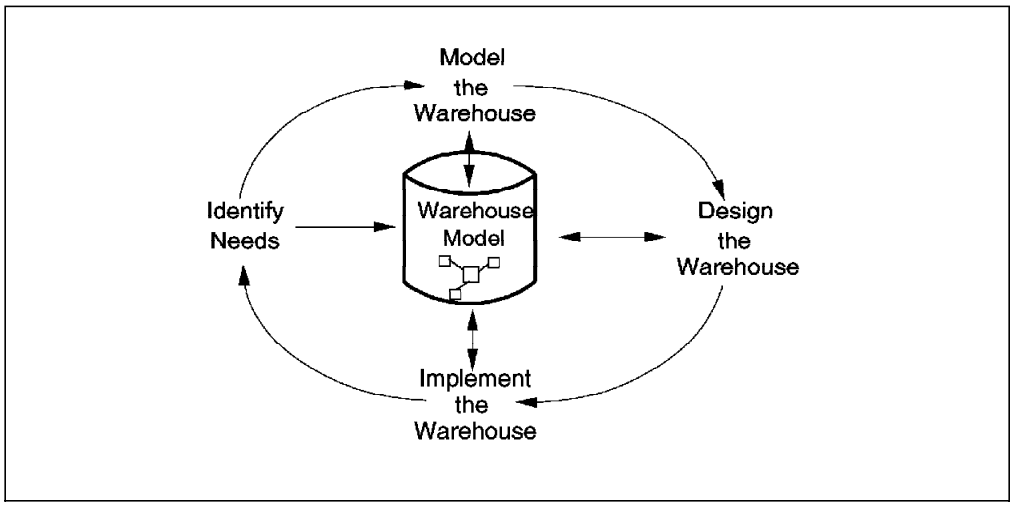


Figure 36. Use of the Warehouse Model throughout the Life Cycle

Chapter 8. Data Warehouse Modeling Techniques

Data warehouse modeling is the process of building a model for the data that is to be stored in the data warehouse. The model produced is an abstract model, and in this sense, it is a representation of reality, or at least a part of reality which the data warehouse is assumed to support.

When considered like this, data warehouse modeling seems to resemble traditional database modeling, which most of us are familiar with in the context of database development for operational applications (OLTP database development). This resemblance should be considered with great care, however, because there are a number of significant differences between data warehouse modeling and OLTP database modeling. These differences impact not only the modeling process but also the modeling techniques to be used.

In Chapter 7, “The Process of Data Warehousing,” the basic issues and steps of a data warehouse modeling process were described. This chapter focuses entirely on the techniques involved in a data warehouse modeling process. It extends and complements Chapter 7, “The Process of Data Warehousing” in several ways:

- Whereas Chapter 7, “The Process of Data Warehousing” focuses on the modeling process, this chapter focuses on data warehouse modeling techniques.
- Whereas Chapter 7, “The Process of Data Warehousing” in large part deals with the basic issues of dimensional modeling and illustrates how end-user requirements can be captured and somehow formalized in what is an initial dimensional model, this chapter investigates data warehouse modeling beyond these aspects.
- Whereas Chapter 7, “The Process of Data Warehousing” considers data warehouse modeling primarily from the point of view of rapid development of an independent data mart, this chapter is concerned with making data warehouse models that are suitable for integration with other data marts or can be deployed within a corporate data warehouse environment.

  Although we focus on modeling techniques, we try to present the techniques as part of a structured approach to data warehouse modeling. However, more work is required in this area to develop a methodological approach to data warehouse modeling.

8.1 Data Warehouse Modeling and OLTP Database Modeling

Before studying data warehouse modeling techniques, it is worthwhile investigating the differences between data warehouse modeling and OLTP database modeling. This will give you a better idea of why new or adapted techniques are required for performing data warehouse modeling will help you understand how to set up a data warehouse modeling approach or methodology.

8.1.1 Origin of the Modeling Differences

There are three main reasons why data warehouse modeling requires modeling techniques other than OLTP database modeling or why traditional modeling techniques, when used in data warehouse development projects, require a significantly different focus.

- A data warehouse has base properties that make it fundamentally different from OLTP databases. In the next section, these properties and the impact they have on data warehouse modeling are investigated further.
- The computing context in which a data warehouse resides differs from the context in which OLTP databases reside. Users of OLTP applications are ″shielded″ from the database structure because they interact through user interfaces and use application services for working with the databases. Users of a data warehouse, however, are much more directly involved with the data warehouse model and the way data is organized in the warehouse. Failing to make models that are simple to understand and directly represent the end user′s perception of reality is one of the worst things that can happen to a data warehouse enablement project.
- Inherent to data warehouse enablement is the fuzziness and incompleteness of end-user requirements and the continuous evolution of the data warehouse. These incomplete requirements call for a flexible modeling process and for techniques which are appropriate for evolutionary development. The risks of flexible and evolutionary software development are incoherence and inconsistency of the end result. These issues certainly require attention when performing data warehouse modeling.

Most of the above reasons for why data warehouse modeling is different from OLTP database modeling also apply in the context of data mart development. Although the development of data marts may appear to be less complicated than the development of corporate data warehouses, many of the properties of a data warehouse that make modeling so different from OLTP database modeling also apply for data mart development projects. In addition, the impact of end users and end-user requirements on the modeling process and techniques applied for data marts become even more important for data warehouses.

8.1.2 Base Properties of a Data Warehouse

Some of the most significant differences between data warehouse modeling and OLTP database modeling are related to the base properties of a data warehouse, which are summarized in Figure 37 on page 83.

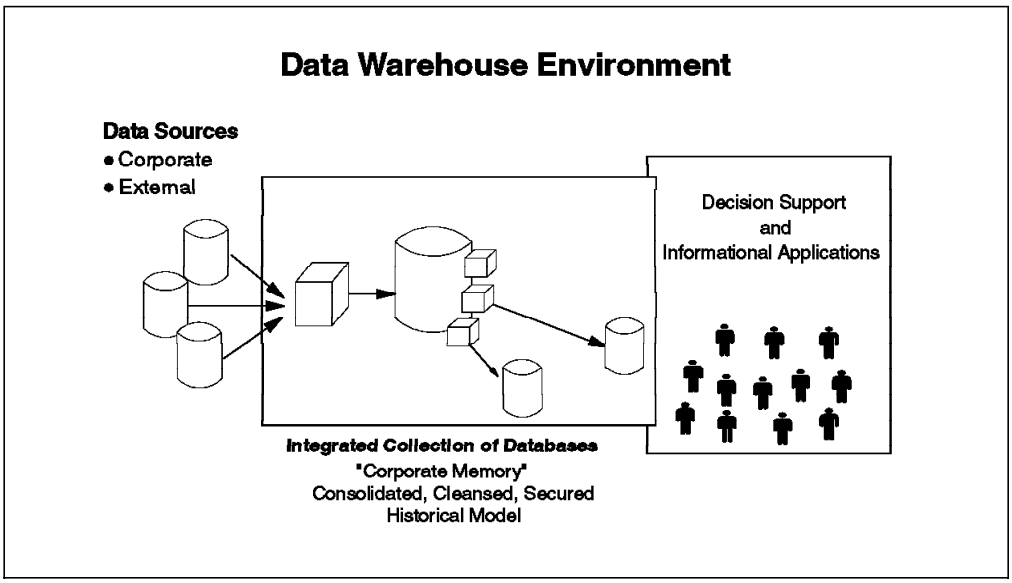


Figure 37. Base Properties of a Data Warehouse.

A data warehouse is an integrated collection of databases rather than a single database. It should be conceived as the single source of information for all decision support processing and all informational applications throughout the organization. A data warehouse is an organic ′thing′, and it tends to become big, if not big from the beginning. In addition to the obvious requirement that a data warehouse should satisfy the needs of end users, there is also a great need to achieve maximum consistency throughout the whole data warehouse environment, at the level of primitive data and derived data, and also within the information derivation processes themselves.

A data warehouse contains data that belongs to different information subject areas, which can be the basis for logically partitioning the data warehouse in several different (conceptual or even physical) databases. A data warehouse also contains different categories of data. It contains primitive data (the ″System of Record″) either represented and organized as an accumulation of captured source data changes, business events and transactions, or as an interpreted and well-structured historical database. In many cases both representations of primitive data are present in the data warehouse and are positioned and mapped to form an integrated collection of data that represents ″the corporate memory.″ Another major category of data in the data warehouse is that which is condensed and aggregated in information analysis databases having a format and layout that is directly suitable for end users to interpret and use. A data warehouse also usually contains ″support databases,″ which are not directly of interest to end users for their data analysis activities but are important components in the process of capturing source data and delivering consistent information to end users.

Clearly, data warehouse modeling must consist of different kinds of modeling techniques. The System of Record is usually best if not modeled using the same modeling techniques as the end-user-oriented information analysis databases. If, in addition, one considers that end users may be dealing with decision support tools (query and reporting, OLAP, data mining, ...) and informational applications that have different usage and development characteristics, it becomes clear that data warehouse modeling is in fact a compilation of different modeling techniques, each with its own area of applicability.

8.1.3 The Data Warehouse Computing Context

Data warehouses have to be developed within a computing context that differs from the context in which OLTP database applications are developed (see Figure 38).

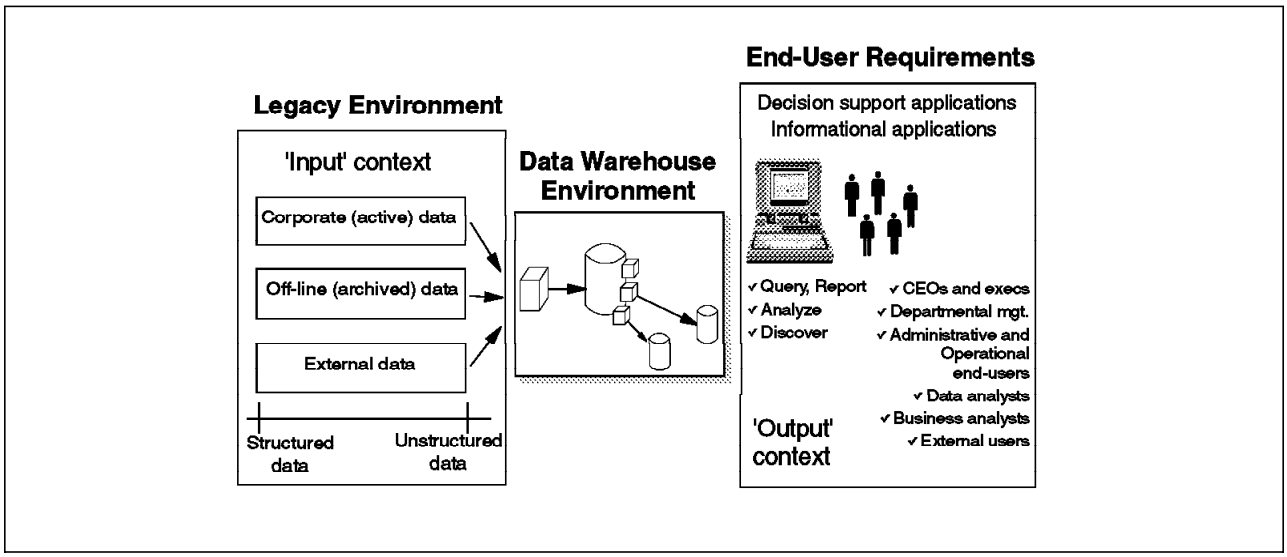


Figure 38. Data Warehouse Computing Context.

There is a fundamental difference between the way end users use OLTP databases and data warehouses. OLTP users are shielded from the databases by an application layer. They perform tasks, usually consisting of a fixed number of predefined database operations, which are part of a fixed transaction workflow.

Data warehouse applications are totally different. They are data-centric rather than process-centric. End users deal almost directly with the data and there are no fixed workflows (with a few exceptions here and there). End users are not interested in recording data in the warehouse: they want to get information out of the warehouse. They raise questions against the warehouse, test and verify hypotheses with information they drag out of the warehouse, reconstruct chains of events, which they then analyze possibly to detect patterns or seasonal trends, and make extrapolations and projections for the future.

Data warehouses are very much open collections of data, and end-user involvement in the enablement process is known to be a vital element of success. In addition, good data warehouses realize what could be called the *information supermarket principle*, whereby end users freely access the data warehouse when they need information for their own purposes.

Figure 38 also points to the other side of the coin. Data warehouse developers, including those who do data warehouse modeling, do have to take into account that the data warehouse ″input context″ consists of a legacy data processing environment residing in a legacy business process environment. Required data may not be available or perhaps cannot be captured at the sufficient level of detail, unless money and effort are spent changing the legacy input environment. Data warehouse enablement projects therefore often get involved with business process and source application reengineering.

All of this has a fundamental impact on the modeling process as well as on the techniques used for producing the data warehouse model, including the ″bag of tricks″ (heuristics, guidelines, metrics, etc.) the data warehouse modeler uses to make things happen the way they should happen.

8.1.4 Setting Up a Data Warehouse Modeling Approach

One of the real challenges that a data warehouse modeling expert faces is to combine suitable modeling techniques in an approach that is end-user focused, leads to a logically integrated historical data organization, supports the delivery of consistent information to end users, and is flexible and scalable.

The main requirements for a solid data warehouse modeling approach can be summarized as follows:

- It has to incorporate several different modeling techniques in a well-balanced and integrated approach. In this chapter, we investigate a number of modeling techniques that we believe are core techniques for data warehouse modeling.
- Each modeling technique should have its own area of applicability. Obviously, the fewer techniques that are blended and integrated into a modeling process, the easier the process becomes. Modeling techniques with a broad scope of applicability therefore are highly recommended. This chapter should help you in selecting suitable modeling techniques and recognizing their scope of applicability.
- It is of the utmost importance that end users see a single, well-integrated, and simple-to-interpret logical model of the data warehouse. This logical model is one of the centerpieces of the data warehouse metadata. Simplicity for the end user and integration and consolidation of the historical data are key principles that the modeling approach should help provide.
- Recalling that a data warehouse environment is an organic thing and that fuzziness and incompleteness are inherent characteristics of data warehouse enablement, any approach to data warehouse modeling should be flexible and provide support for an evolutionary data warehouse development process. End users must be involved maximally in the modeling process itself. Therefore, the modeling techniques and the results they produce must be understandable for information analysts who have, by definition, no technical IT background. Knowing that, in addition, flexibility and support for evolutionary development call for the support of constant changes and extensions applied to the data warehouse model. However, providing this flexibility in setting up a data warehouse modeling approach is very much a challenge.

Tools can have a significant impact on the establishment of a data warehouse modeling approach for an organization. Data modeling tools and metadata catalogs are important for the data warehouse modeling approach. They usually have a significant impact on the choice of modeling techniques.

Although it is not the intention of this chapter to fully describe a data warehouse modeling approach, we do want to contribute to the establishment of a realistic and well-structured data warehouse modeling approach. In the next section, we present a survey of the most important techniques that should somehow be incorporated in an overall modeling approach. At the end of this chapter, we bring the different elements of the ″modeling puzzle″ together and consider how to set up a data warehouse modeling approach.

8.2 Principal Data Warehouse Modeling Techniques

Listed below are the principal modeling techniques (beyond what can be considered ″traditional″ database modeling techniques, such as ER modeling and normalization techniques) that should be arranged into an overall data warehouse modeling approach.

1. Dimensional data modeling
2. Temporal data modeling
3. Techniques for building generic and reusable data models (sometimes referred to as pattern-oriented data modeling). These techniques are much more extensively and frequently considered in the context of software development. Data warehouse modelers should learn to apply some of these techniques, although a transposition from a software development context to a data warehouse development context may not always be obvious.
4. Data architecture modeling consists of a combination of top-down enterprise data modeling techniques and bottom-up (detailed) model integration. Data architecture modeling also should provide the techniques for logical data partitioning, granularity modeling, and building multitiered data architectures.

Other modeling techniques may have to be added to the overall approach. If, for example, the data warehouse also incorporates multimedia data types such as documents and images or if end-users are involved in other types of informational applications than dimensional data analysis.

To keep the scope and complexity of this chapter within realistic boundaries, we concentrate our attention on dimensional data modeling and temporal data modeling. We present the base techniques for dimensional and temporal data modeling, and, in the course of the discussion, we comment on techniques for building generic and reusable data models. Be aware that much more can be said about dimensional and temporal data modeling than what we say in this chapter and that we only scratch the surface of the techniques and approaches for building generic and reusable data models.

Data architecture modeling and advanced modeling techniques such as those suitable for multimedia databases and statistical databases are beyond the scope of this chapter (and as a matter of fact, beyond the scope of this book).

8.3 Data Warehouse Modeling for Data Marts

Data marts can loosely be defined as data warehouses with a narrower scope of use. Data marts are focused on a particular data analysis business problem or on departmental information requirements (Figure 39 on page 87 illustrates this).

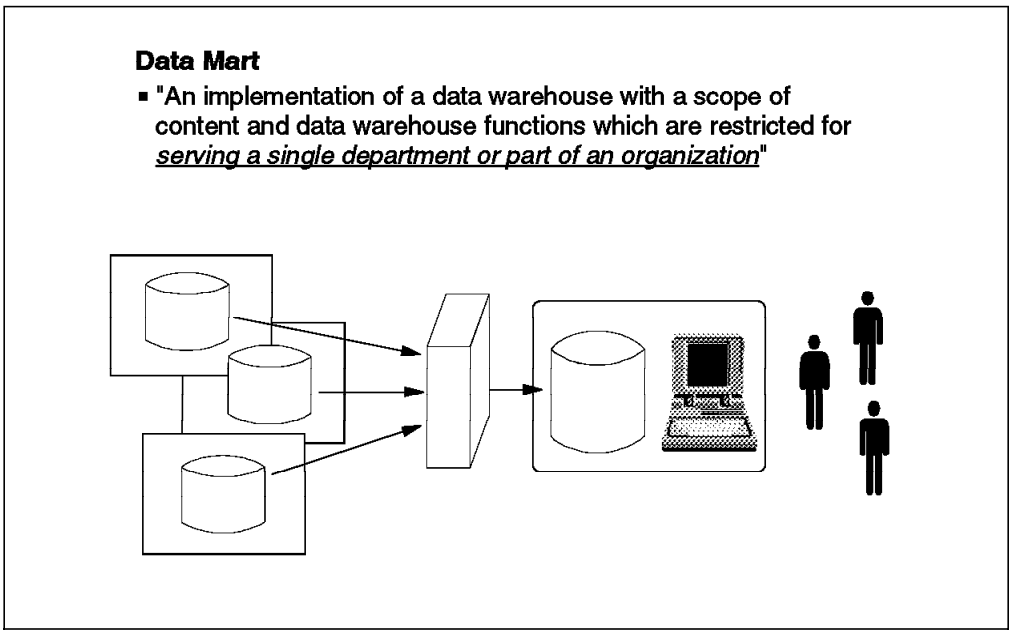


Figure 39. Data Marts.

Some of the complexities inherent in data warehouses are usually not present in data-mart-oriented projects. Techniques and approaches for data mart development are somewhat different from those applied for data warehouse development. Data architecture modeling, for instance, which is a crucial technique for data warehouse development, is far less required for data marts.

The complexities of data warehousing usually make data mart development, and modeling in particular, appear to be less complicated and time consuming. In reality, this is true. But you should refrain from concluding that data mart development is by definition simple and easy going. We have seen many cases of data mart solutions based on quick and dirty development using an OLAP tool that end users and the data mart administrator apparently thought was good for the information analysis that had to be performed. Such solutions usually do not last very long. Thus, we advocate that in data mart development a high level of attention be given to proper data modeling.

Modeling for the data mart has to be more end-user focused than modeling for a data warehouse. End users must be involved in the data mart modeling process, as they obviously are the ones who will use the data mart. Because you should expect that end users are not at all familiar with complex data models, the modeling techniques and the modeling process as a whole should be organized such that complexity is transparent to end users.

This chapter has a significant focus on dimensional data modeling. As a matter of fact, dimensional data modeling is a powerful data modeling technique for data mart development. When the data mart is developed to support a dimensional data analysis business problem (such as when users perform OLAP activities using the information in the data mart), dimensional data modeling is by far advisable as the modeling technique to use. The dimensional modeling techniques presented in this chapter therefore are also suitable for those who are primarily interested in a narrower scope of the work of data warehouse modeling, that is, those who are interested in developing data models for data marts. We use the term *data warehouse modeling* throughout the chapter, however, whether the modeling is done in the context of a data warehouse or in

the context of a data mart. Where relevant distinctions between data warehouse and data mart modeling are present, we indicate this in the text.

You should not conclude from the above that we advocate dimensional modeling for data warehouses. The suitability of dimensional modeling for producing data models with a very large scope of coverage is intensely debated among today's experts. At the end of the chapter, after all the techniques for data warehouse modeling have been presented, we will be in a better position to evaluate that ongoing discussion.

8.4 Dimensional Modeling

The approach to dimensional modeling developed in this chapter is summarized in Figure 40 on page 89. The focus of our discussion will primarily be on data modeling requirements. In subsequent sections of this chapter, the various steps in the dimensional modeling process are investigated in more detail.

A debate raging in the data warehouse modeling world today is that between traditionalists who promote ER modeling and third normal form as the only good modeling approach for data warehouse modeling, and those who proclaim that ER models are unusable for data warehouse modeling because they are too technical and too complex for end users. For these modeling experts, dimensional modeling provides "salvation," the promised land of dimensional modeling being, primarily, the techniques that produce flattened dimensional models (star models, as opposed to snowflake models, that show the structure of the dimension hierarchies in the dimensional model).

We try to stay out of this debate as much as possible, although staying out of it completely is impossible.

In this chapter, we emphasize that models should be produced that represent the problem domain as directly as possible. Technical considerations and design techniques should be introduced only when the (conceptual) model is complete.

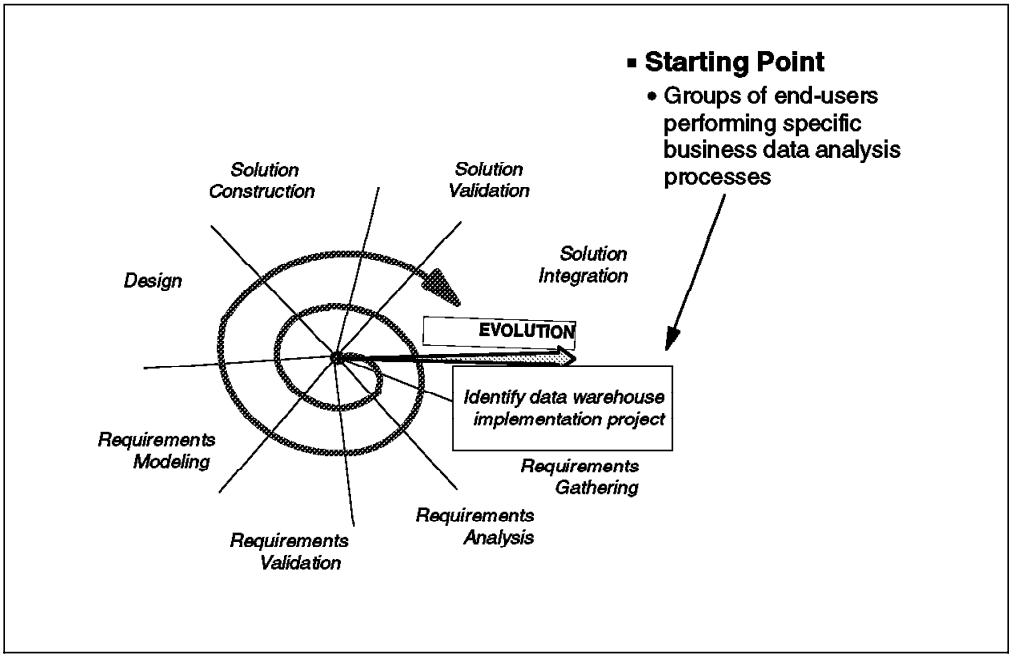


Figure 40. Dimensional Modeling Activities.

The following list introduces the requirements topics, which are discussed in more detail in subsequent sections of this chapter.

- **Requirements Gathering**. During requirements gathering, end-user requirements are collected and documented. Requirements gathering is often incorporated in some way into studies of the business processes and information analysis activities in which end users are involved. Requirements gathering therefore is very much oriented toward understanding the problem domain for which the modeling will be done. Usually, end-user requirements at this stage are documented rather informally or at least they are not represented in detailed schemas. Techniques for requirements gathering include very traditional techniques such as interviews with end users, study of existing documents and reports, and monitoring the ongoing information analysis activities. Experience with business process engineering and information analysis in itself usually contributes significantly in this stage. The results of requirements gathering are used as the basis for producing the dimensional models.
- **Requirements Analysis**. During requirements analysis, informal end-user requirements are further investigated, and initial dimensional models are produced showing facts, measures, dimension keys, and dimension hierarchies. Dimension hierarchies can include parallel hierarchical paths. Models produced during requirements analysis must be kept simple, because these initial dimensional models must be discussed with end users. Figure 41 on page 90 shows a sample schematic notation technique that can be used for requirements analysis. A schema like that in Figure 41 on page 90 is what we will call the *initial dimensional model of the problem domain*.

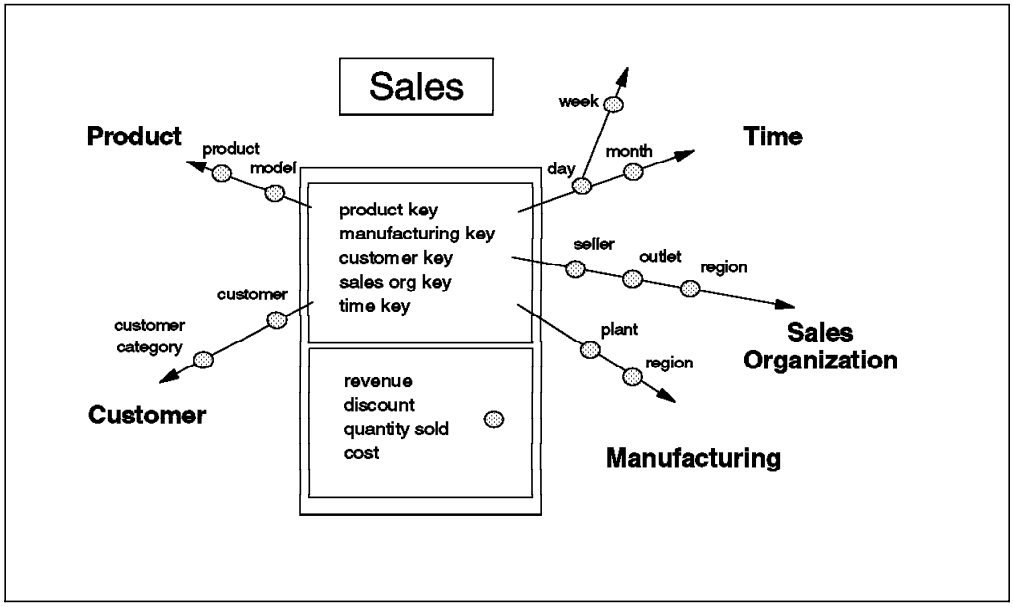


Figure 41. Schematic Notation Technique for Requirements Analysis.

The activities that are part of requirements analysis are illustrated in Figure 42. While the initial dimensional models are being produced, the business directory definitions for facts, measures, and dimensions and all other elements in the model should be established. These definitions are the core of the business metadata that end users need when using the data warehouse for their information analysis activities.

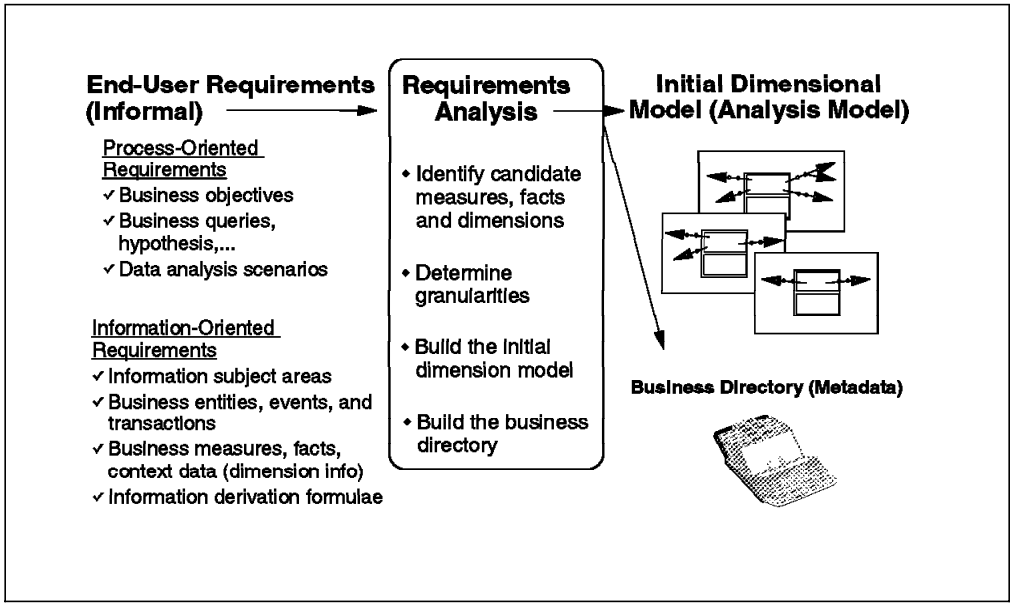


Figure 42. Requirements Analysis Activities.

- **Requirements Validation**. Initial dimensional models are used in the process of validating the end-user requirements and for assessing the scope and impact of the development project. These activities are schematically summarized in Figure 43 on page 91.

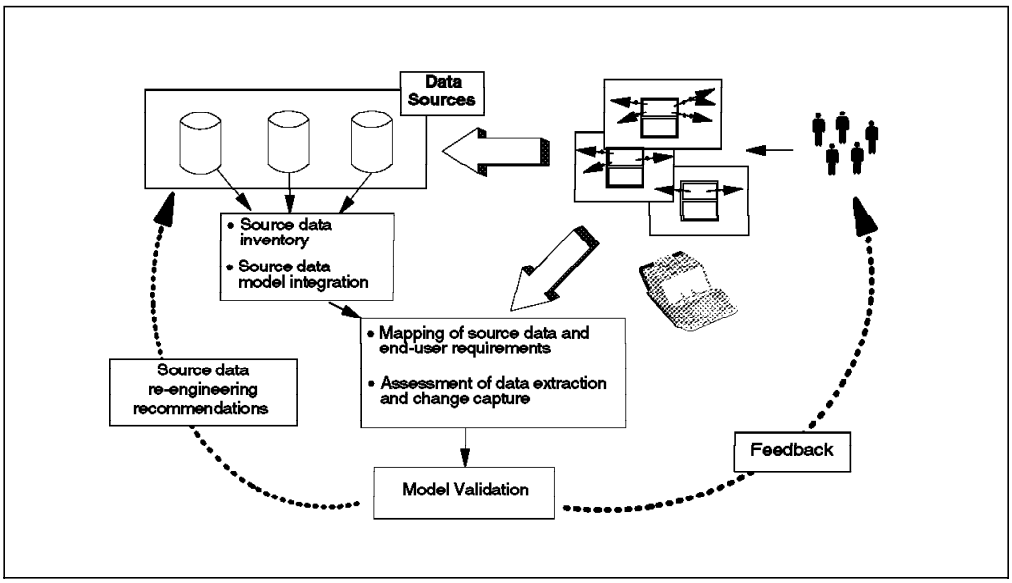


Figure 43. Requirements Validation.

- **Requirements Modeling**. Validated initial models are further developed into detailed dimensional models, showing all elements of the model and their properties. Detailed dimensional models can further be extended and optimized. Many techniques in this area should be thought of as advanced modeling techniques. Not every project requires all of them to be applied. We cover some of the more commonly applied techniques and indicate what other issues may have to be addressed. The major activities that are part of requirements modeling are illustrated in Figure 44.

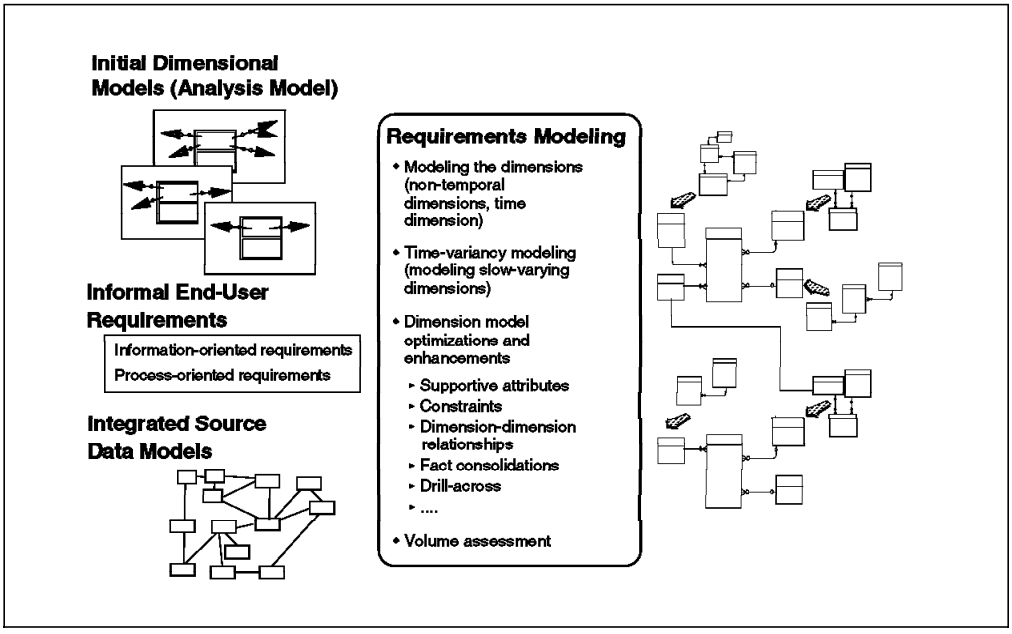


Figure 44. Requirements Modeling.

When advanced dimensional modeling techniques are used such as the ones indicated in Figure 44, the dimensional model usually tends to become complex and dense. This may cause problems for end users. To solve this, consider building two-tiered data models, in which the back-end tier comprises all of the model artifacts and the full structure of the model,

whereas the front-end tier (the part of the model with which the end user is dealing directly) is a derivation of the entire model, made simple enough for end users to use in their data analysis activities. Two-tier data modeling is not required as such. If end users can fully understand the dimensional model, the additional work of constructing the two tiers of the model should not be done.

- **Design, Construction, Validation, and Integration**. Once requirements are modeled, possibly in a two-tiered dimensional model, design and construction activities are to be performed. These will further extend and possibly even change the models produced in the previous stages of the work, to make the resulting solution implementable in the software infrastructure of the data warehouse environment. Also, a functional validation of the proposed solution must be performed, together with the end users. This usually results in end users using the constructed solution for a while, giving them the opportunity to work with the information that has been made available to them in a local solution (perhaps in a data mart). In addition, the local solution may then be integrated into a more global data warehouse architecture, including the model of the data produced.

  We attach particular importance to clearly separating modeling from design. Good modeling practice focuses on the essence of the problem domain. Modeling addresses the ″what″ question. Design addresses the question of ″how″ the model representing reality has to be prepared for implementing it in a given computing environment.

  The separation between modeling and design is of significant importance for data warehouse modeling. Unfortunately though, all too often modeling issues are mixed with design issues, and, as a consequence, end users are confronted with the results of what typically are design techniques. Because modeling is not always already separated from design, many data warehouse models have a technical outlook.

  Neglecting a clear separation between modeling and design also results in models that are closely linked with the computing environment in general and with tools in particular. Thus it is difficult to integrate the models with others and adapt and expand them. Keep in mind that a data warehouse and data warehouse models are very long lasting.

Each of the requirements steps in the dimensional modeling process are now discussed in more detail. The design, construction, validation, and integration steps are discussed within the context of the dimensional modeling requirements.

8.4.1 Requirements Gathering

End-user requirements suitable for a data warehouse modeling project can be classified in two major categories (see Figure 45 on page 93): **process-oriented requirements**, which represent the major information processing elements that end users are performing or would like to perform against the data warehouse being developed, and **information oriented requirements**, which represent the major information categories and data items that end users require for their data analysis activities.

Typically, requirements can be captured that belong to either or both of these categories. The types of requirements that will be available and the degree of precision with which the requirements will be stated (or can be stated) often depend on two factors: the type of information analysis problem being

considered for the data warehouse implementation project, and the ability of end users to express their information needs and the scenarios and strategies they use in their information analysis activities.

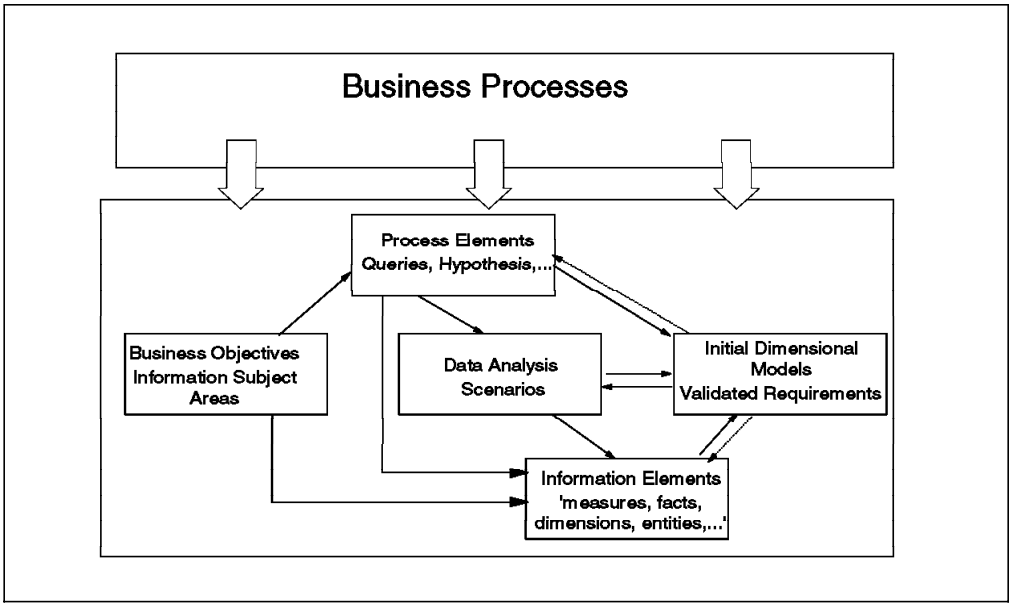


Figure 45. Categories of (Informal) End-User Requirements.

8.4.1.1 Process Oriented Requirements

Several types of process-oriented requirements may be available:

- **Business objectives**

  Business objectives are high-level expressions of information analysis objectives, expressed in business terms. One or more business objectives can be specified for a given data warehouse implementation project.

  As an example, in the CelDial case study (see Appendix A, “The CelDial Case Study” on page 163), the business objectives could be stated as:

  – ″The data warehouse has to support the analysis of manufacturing costs and sales revenue of products manufactured and sold by CelDial.″

    The combined business objectives can be used in the data warehouse implementation project as indicators of the scope of the project. They can also be used to identify information subject areas involved in the project and as a means to identify (usually high-level) measures of the business processes the end user is analyzing. In the CelDial example, the apparent information subject areas are Products and Sales. The objectives indicate that the global measures used in the information analysis process are ″manufacturing cost″ and ″sales revenue.″ Notice that these high-level measures ″hide″ a substantial requirement in terms of detailed data to calculate them.

- **Business queries**

  Business queries represent the queries, hypotheses, and analytical questions that end users issue and try to resolve in the course of their information analysis activities. Just as with business objectives, business queries are expressed in business terms. You should expect that they are

usually not precisely formulated. They are certainly not expressed in terms of SQL.

Examples of frequently encountered categories of business queries are:

- Existence checking queries, such as ″Has a given product been sold to a particular customer?″
- Item comparison queries, such as ″Compare the value of purchases of two customers over the last six months,″ or ″Compare the number of items sold for a given product category, per store, and per week.″
- Trend analysis queries, such as ″What is the growth in item sales for a given set of products, over the last 12 months?″
- Queries to analyze ratios, rankings, and clusters, such as ″Rank our best customers in terms of dollar sales over the last year.″
- Statistical analysis queries, such as ″Calculate the average item sales per product category, per sales region.″

For the CelDial case study, several business queries were identified. For the sake of this chapter, we selected three of them to use for illustration:

- (Q1) What is the average quantity on hand this month, for each product model in each manufacturing plant?
- (Q2) What is the total cost and revenue for each model sold today, summarized by outlet, outlet type, region, and corporate sales levels?
- (Q3) What is the total cost and revenue for each model sold today, summarized by manufacturing plant and region?

For a complete description of the CelDial case study, see Appendix A, “The CelDial Case Study” on page 163 and the description of the modeling process in Chapter 7.

- **Data analysis scenarios**

  Data analysis scenarios are a good way of adding substance to the set of requirements being captured and analyzed. Unfortunately, they are more difficult to obtain than other processing requirements and thus are not always available for requirements analysis.

  Essentially two types of data analysis scenarios are of interest for data warehouse modeling:

  - **Query workflow scenarios:** These scenarios represent sequences of business queries that end users perform as part of their information analysis activities. Query workflow scenarios can significantly help create a better understanding of the information analysis process.
  - **Knowledge inference strategies**: These end-user requirements acknowledge the fact that activities performed by end users of a data warehouse have expert system characteristics. As with query workflow scenarios, these strategies can provide more understanding of the activities performed by end users. The simplest forms of knowledge inference strategies are those that show how users roll up and drill down along dimension hierarchies.

    Whether or not these end-user requirements will be available depends on the capabilities of end users to express how they get to an answer or find a solution for their problems as well as on the type of data warehouse application that is being considered for the modeling project.

8.4.1.2 Information-Oriented Requirements

Information-oriented requirements capture an initial perception of the kinds of information end users use in their information analysis activities. There are different categories of information-oriented requirements that may be of interest for the requirements analysis and data warehouse modeling process:

- **Information subject areas**

  Information subject areas are high-level categories of business information. Information subject areas usually are used to build the high-level enterprise data model. When available, information subject areas indicate the scope of the data warehouse project. They also contribute to the requirements analyst's ability to relate the data warehouse project with other (already developed) parts of the data warehouse or to data marts.

For the CelDial case study, the information subject areas of interest are: Products, Sales (including Sales Organization), and Manufacturing (including Inventories). Whether or not the Customers information subject area is present in the scope of the CelDial case study is debatable. Although customer sales are involved, there is no apparent substantial requirement that indicates that the Customers subject area should also be included in the project. In addition, if retail outlets within the Sales Organization also hold inventories of products they may sell, then most probably Inventories should become an information subject area in its own right rather than be incorporated in Manufacturing. Debates such as these are typical when trying to establish the information subject areas involved in a data warehouse development project.

- **High-level data models, ER and/or dimensional models**

  Several data models may be available and could be used to further specify or support end-user requirements. They can be available as high-level enterprise data models, ER models, or dimensional models. The ER models may be collected by reengineering and integrating source data models. Dimensional models may be the result of previous dimensional data warehouse modeling projects.

  Figure 46 on page 96 illustrates the relationships among the various data models in the data warehouse modeling process.

  In user-driven modeling approaches, source data models are used as aids in the process of fully developing the data warehouse model.

  Source data models may have to be constructed by using reverse engineering techniques that develop ER models from existing source databases. Several of these models may first have to be integrated into a global model representing the sources in a logically integrated way.

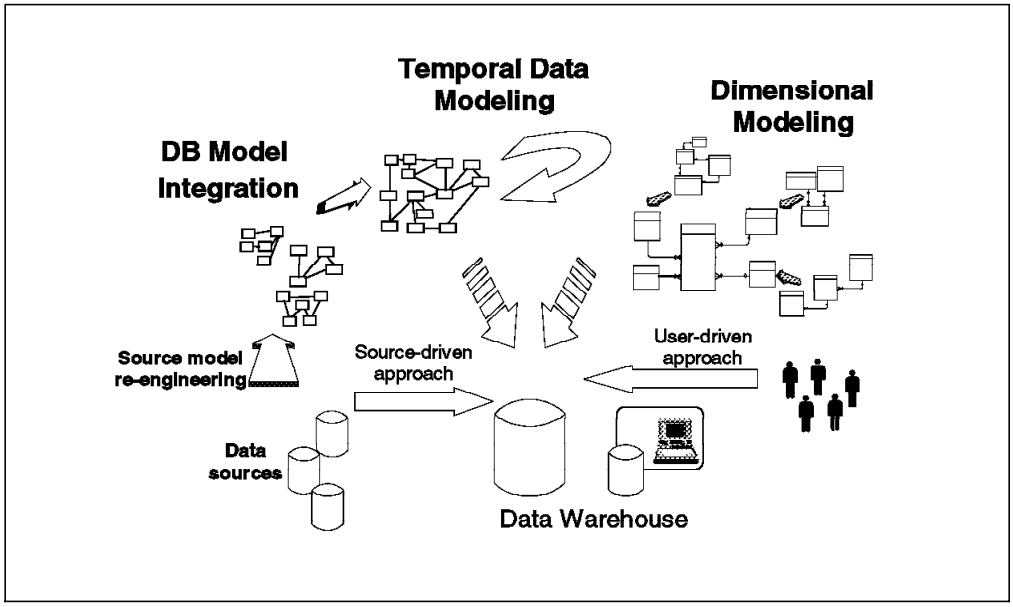


Figure 46. Data Models in the Data Warehouse Modeling Process.

8.4.2 Requirements Analysis

Requirements analysis techniques are used to build an initial dimensional model that represents the end-user requirements captured previously in an informal way. The requirements analysis produces a schematic representation of a model that information analysts can interpret directly. The results of requirements analysis will be the primary input for data warehouse modeling once they have passed the requirements validation phase.

The scope of work of requirements analysis can be summarized as follows:

- Determine candidate measures, facts, and dimensions, including the dimension hierarchies.
- Determine granularities.
- Build the initial dimensional model.
- Establish the business directory for the elements in the model.

Figure 47 on page 97 summarizes the context in which initial dimensional modeling is performed and the kinds of deliverables that are produced.

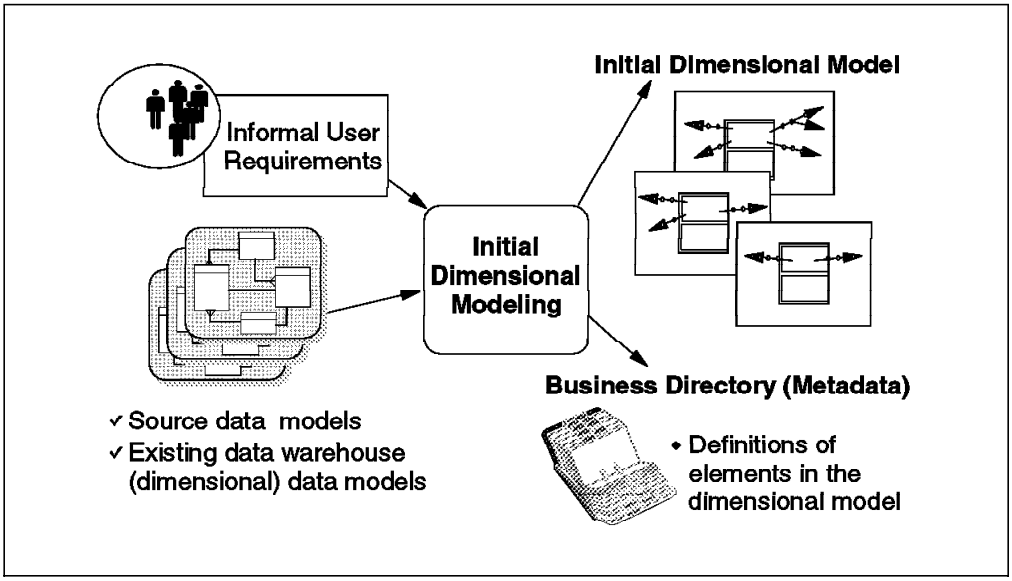


Figure 47. Overview of Initial Dimensional Modeling.

Figure 48 illustrates a notation technique that can be used to schematically document the initial dimensional model. It shows facts (or fact tables, if you prefer) with the measures they represent and the dimension hierarchies or aggregation paths associated with the facts. Dimension hierarchies are represented as arrows showing intermediary aggregation points. The dimensions may include alternate or parallel dimension hierarchies. Dimension hierarchies are given names drawn from the problem domain of the information analyst. These initial dimensional models also formally state the lowest level of detail—the granularity—of each dimension. An initial dimensional model consists of one or more such schemas.

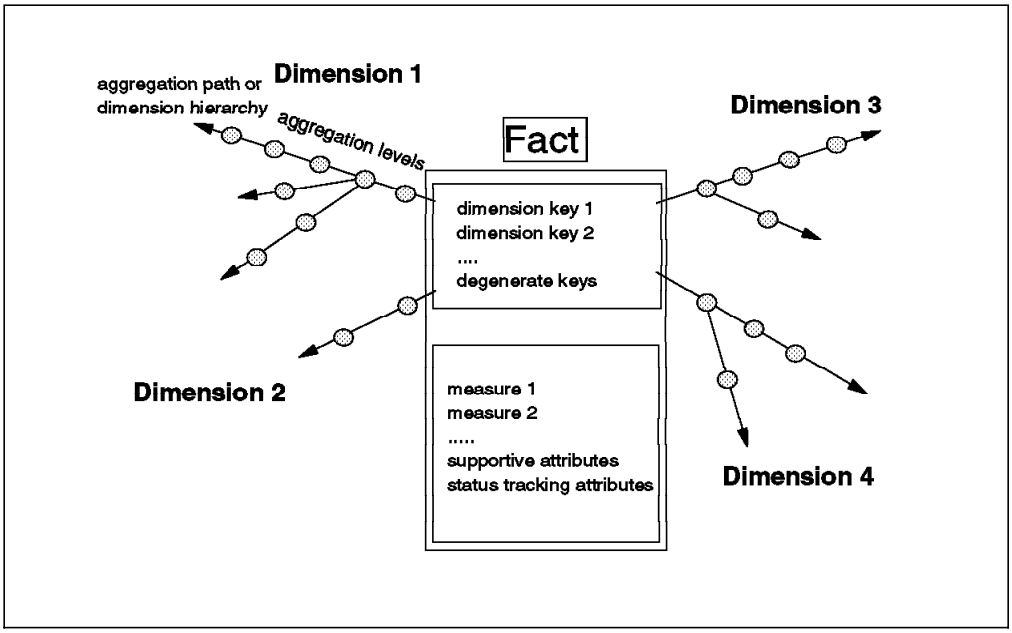


Figure 48. Notation Technique for Schematically Documenting Initial Dimensional Models.

8.4.2.1 Determining Candidate Measures, Dimensions, and Facts

To build an initial dimensional model, the following base elements have to be identified and arranged in the model:

- Measures
- Dimensions and dimension hierarchies
- Facts

Several approaches can be used to determine the base elements of a dimensional model. In reality, analysts combine the use of several of the approaches to find appropriate candidate elements for the model and integrate their findings in an initial dimensional model, which then combines several different views on reality. Because the requirements analysis process is nonlinear and knowing that inherent relationships exist between the candidate elements, it does not really matter which approach is used, as long as the process is performed with a clear perspective on the business problem domain.

The approaches essentially differ in the sequence with which they identify the modeling elements. Some of the most common approaches are:

- Determine measures first, then dimensions associated with measures, then facts

  This approach could be called the *query-oriented approach* because it is the approach that flows naturally when the requirements analyst picks up the end-user queries as the first source of inspiration. Chapter 7, “The Process of Data Warehousing” on page 49 and the case study in Appendix A, “The CelDial Case Study” on page 163 were developed by using this approach.

- Determine facts, then dimensions, then measures

  This approach is a *business-oriented* approach. Typically, it tries to determine first the fundamental elements of the business problem domain (facts and measures) and only then are the details required by the end users developed in it. This chapter shows how this approach can be used to compensate the strict end-user-oriented view when trying to develop more fundamental and longer lasting models for the problem domain.

- Determine dimensions, then measures, then facts

  This approach frequently is used when the source data models are being used as the basis for determining candidate elements for the initial dimensional model. We refer to it as the *data-source-oriented* approach.

Notice that facts, dimensions, and measures determined during this stage are *candidate* elements only. Some of them may later disappear from the model, be replaced by or merged with others, be split in two or more, or even change their ″nature.″

Candidate Measures: Candidate measures can be recognized by analyzing the business queries. Candidate measures essentially correspond to data items that the users use in their queries to measure the performance or behavior of a business process or a business object.

For the CelDial project, the following candidate measures are present in Q1, Q2 and Q3:

- Average quantity on hand
- Total Cost
- Total Revenue

For a complete list of measures, refer to Chapter 7, “The Process of Data Warehousing” on page 49 and Appendix A, “The CelDial Case Study” on page 163.

Determining candidate measures requires smart, not mechanical, analysis of the business queries. Good candidate measures are numeric and are usually involved in aggregation calculations, but not every numeric attribute is a candidate measure. Also, candidate measures identified from the available queries may have peculiar properties that do not really make them ″good″ measures. We investigate some properties of measures later in this chapter and indicate how they may affect the model.

Measure Granularities within a Dimensional Model. The granularity of a measure can be defined intuitively as the lowest level of detail used for recording the measure in the dimensional model. For instance, Average Quantity On Hand can be considered to be present in the model per day or per month. Average Quantity On Hand could also be considered at the level of detail of product or perhaps at product category level or packaging unit.

Measures are usually associated with several dimensions. The granularity of a measure is determined by the combination of the recording details of all of its dimensions.

Different measures can have identical granularities. Because both Total Cost and Total Revenue seem to be associated with sales transactions in the CelDial case, they have identical granularities. We show next that measures with identical granularities are candidates for being part of another element of the dimensional model: the fact.

Determining the right granularities of measures in the data warehouse model is of extreme importance. It basically determines the depth at which end users will be able to perform information analysis using the data warehouse or the data mart. For data warehouses, the granularity situation is even more complex. Fine granular recording of data in the data warehouse model supports fine detailed analysis of information in the warehouse, but it also increases the volume of data that will be recorded in the data warehouse and therefore has great impact on the size of the data warehouse and the performance and resource consumption of end-user activities. As a base guideline, however, we advocate building initial dimensional models with the finest possible granularities.

Candidate Dimensions: Measures require dimensions for their interpretation. For example, average quantity on hand requires that we know with which product, inventory location (manufacturing plant), and period of time (which day or month) the value is associated. Average quantity on hand for CelDial therefore is to be associated with three dimensions: Product, Manufacturing, and Time. Likewise, Total Revenue analyzed in Query Q2 requires Sales (shorthand for Sales Organization), Product, and Time as dimensions, whereas for Query Q3, the dimensions are Manufacturing, Product, and Time.

Dimensions are ″the coordinates″ against which measures have to be interpreted. Analyzing the query context in which candidate measures are specified results in identifying candidate dimensions for each of the measures, within the given query context. Notice that this happens ″per measure″ and ″per query.″ One of the next steps involves the consolidation of candidate measures and their dimensions across all queries.

For CelDial, four candidate dimensions can thus be identified at this time: Product, Sales Organization, Manufacturing, and Time. The associations between candidate measures and dimensions, for each of the business query contexts of the CelDial case study, are documented in Chapter 7, “The Process of Data Warehousing” on page 49 and Appendix A, “The CelDial Case Study” on page 163.

A more generic and usually more interesting approach for identifying candidate dimensions consists of investigating the fundamental properties of candidate measures, within the context of the business processes and business rules themselves. In this way, dimensions can be identified in a much more fundamental way. Determining candidate dimensions from the context of given business queries should be used as an aid in determining the fundamental dimensions of the problem domain.

As an example, Sales revenue is inherently linked with Sales transactions, which must, within the CelDial business context, be associated with a combination of Product, Sales Organization, Manufacturing and Time. Because Sales transaction also involves a customer (for CelDial, this can be either a corporate customer or an anonymous customer buying ″off the counter″), we may decide to add Customer as another dimension associated with the sales revenue measure.

Candidate Facts: In principle, measures together with their dimensions make up facts of a dimensional model.

Two facts can be identified in the CelDial case: Sales and Inventory. The obvious interpretation of the fact that is manipulated in Q1 is that of an inventory record, providing the Average Quantity On Hand per product model, at a given manufacturing plant (the inventory location) during a period of time (a day or a month). For this reason, we call it the *inventory fact*. Given values for all three dimensions, for instance, a model, a manufacturing plant, and a time period, the existence of a corresponding Inventory fact can be established, and, if it exists, it gives us the value of the corresponding Average Quantity On Hand. The fact manipulated in Q2 and Q3 is called Sales. It incorporates two measures, Total Cost and Total Revenue. Both measures are dependent on the same dimensions.

Semantic Properties of Business-Related Facts. Facts are core elements of a dimensional model. A representative choice of facts, corresponding to a given problem domain, can be an enabler for a profound analysis of the business area the end user is dealing with, even beyond what is requested and expected (and what is consequently expressed in the end-user requirements). A choice of representative, business-related facts can also support the extension of the use of the data warehouse model to other end-user problem domains. Identifying candidate facts through the process of consolidating candidate measures and dimensions is a viable approach but may lead to facts with a ″technical″ nature. We recommend that candidate facts be identified with a clear business perspective.

Facts can indeed represent several fundamental ″things″ related to the business:

- A fact can represent a business transaction or a business event (Example: a Sale, representing what was bought, where and when the sale took place, who bought the item, how much was paid for the item sold, possible discounts involved in the sale, etc.).

- A fact can represent the state of a given business object (Example: the Inventory state, representing what is stored where and how much of it was stored during a given period).
- A fact can also represent changes to the state of a given business object (Example: Inventory changes, representing item movements from one inventory to another and the amount involved in the move, etc.).

Guidelines for Selecting Business-Related Facts. Now we further explore the specific characteristics of these types of business-related facts and how they can be used in dimensional modeling. We recommend that you apply the following guidelines for identifying representative, business-related facts:

- **Guideline 1:** Each fact should have a real-world business equivalent.
- **Guideline** 2: Focus on determining business-related facts that represent either:
  - Business transactions or business events

    or
  - Business objects whose state is of interest for the information analyst

    or
  - Business objects whose state changes are of interest for the information analyst

    Whether or not a state model or a state change model (or both) will be used to represent facts in the dimensional model depends on the interests of the information analyst.
- **Guideline** 3: Each fact should be uniquely identifiable, merely by the existence of its real-life equivalent.
- **Guideline** 4: The granularity of the base dimensions of each fact should be as fine-grained as possible.

These guidelines can be used either to drive the dimensional modeling process or to assess and enhance an initial dimensional model, developed on the basis of business query analysis.

Facts Representing Business Transactions or Business Events. A Sale is an example of a fact representing a business transaction or a business event. If we want to analyze its ″performance,″ measures like Total Cost and Total Revenue have to be associated with it. Clearly, such facts represent ″something that happened which is of interest to the analyst.″ A transaction or an event can belong to the business itself, or it can be an outside transaction or event.

Identifying candidate facts using business-related techniques usually results in the identification of additional measures (see Figure 49 on page 102). As an example, if the Sales fact is identified as the thing that represents Sale transactions, Quantity Sold will almost naturally be added as a fundamental measure.

Differentiating a fact that represents a business transaction from a fact that represents an event can be somewhat obscure. Let′s consider the Sales fact, for instance. Business transactions are supposed to ″make changes happen″ in the business environment. In OLTP applications, transactions associated with business transactions apply changes to the database that correspond to changes in the business environment. We usually want to know the effects of these

changes and therefore we want to be able to measure the effects of the business transaction. This is why facts that represent business transactions have measures associated with them.

If, however, we are only interested in knowing if and when a Sale happened, we would keep record of Sale as a business event rather than a business transaction. Usually, for facts that represent business events, no record is kept of any measures. For the Sales fact, this would imply that we would only be able to identify the Sale as something that happened at a certain moment in time, involving a product, an outlet, and a manufacturing plant (or inventory location). Facts that are associated with business events are sometimes called *factless facts* (a term used in *The Data Warehouse Toolkit* by Ralph Kimball) although with our terminology, it would be better to call them *measureless facts*.

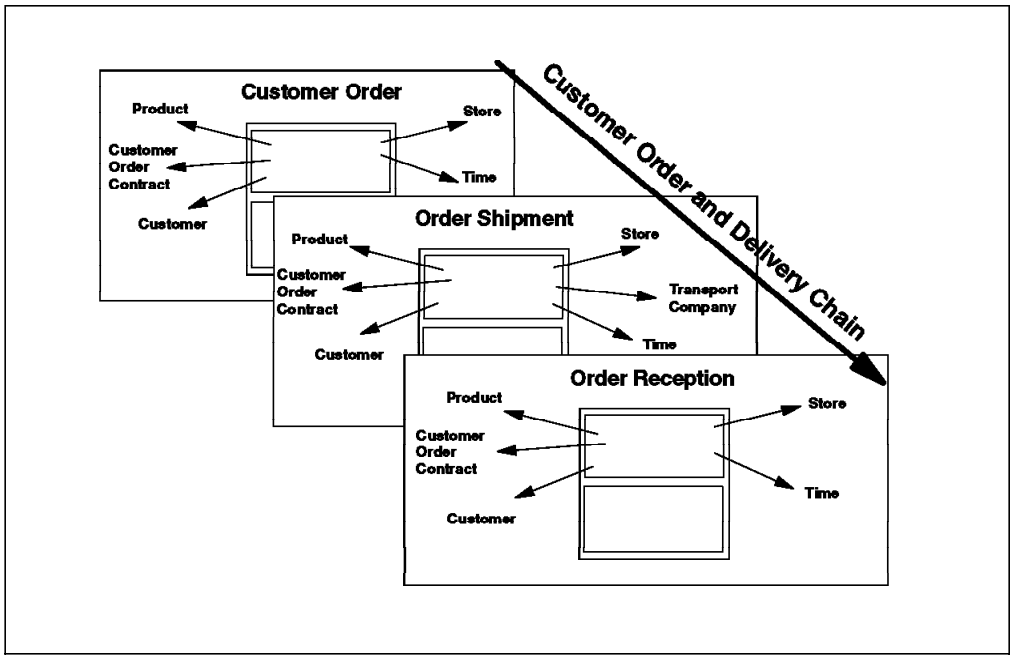


Figure 49. Facts Representing Business Transactions and Events.

Facts Representing the State of Business Objects. The Inventory fact is an example of a business-related fact. It represents the state of the business object ″Inventory,″ which is associated with a product, a manufacturing plant, and a period of time. The state of the Inventory business object is represented here by the measure Average Quantity on Hand.

Notice that the time dimension of the Inventory fact is a duration or a time-period in this case. Measures of the Inventory fact (Quantity on Hand or any other measure associated with this Inventory fact) must represent the state of the particular inventory, during that time period (see Figure 50 on page 103).

The careful reader may at this point have spotted a potential problem: how suitable is the Inventory fact for what end users really want to analyze. Although Inventory very directly represents some of the end-user requirements we are analyzing (notice, we oversimplify the whole situation for the sake of a clear explanation), it does not provide a good solution for analyzing the state of Inventory business objects. One of the basic problems is the time dimension being a duration: if the duration is relatively long with respect to the frequency

with which the Inventory state changes, the Average Quantity On Hand in the Inventory fact is not a very representative measure.

To solve this problem, there are basically three solutions. Either we keep the granularity of the time dimension as it is and add some more measures that give a better (statistical) representation of the Inventory state: We could add Minimum Quantity on Hand, Maximum Quantity on Hand, etc. to try to compensate for the lack of preciseness of Quantity on Hand during the recorded time period. This is not an elegant solution, but it may be the only solution available.

A better solution would be to increase the granularity of our Inventory fact by reducing the representative time duration in the fact: Rather than keep a record of the Inventory state once per month, we could choose to register it per day. In this case, we can basically work with the same measures as before, but we now interpret them on a daily basis rather than monthly. If we decide to solve the problem with the Inventory fact in this way, we obviously have to assess whether the source databases and the data warehouse populating subsystem can support this solution.

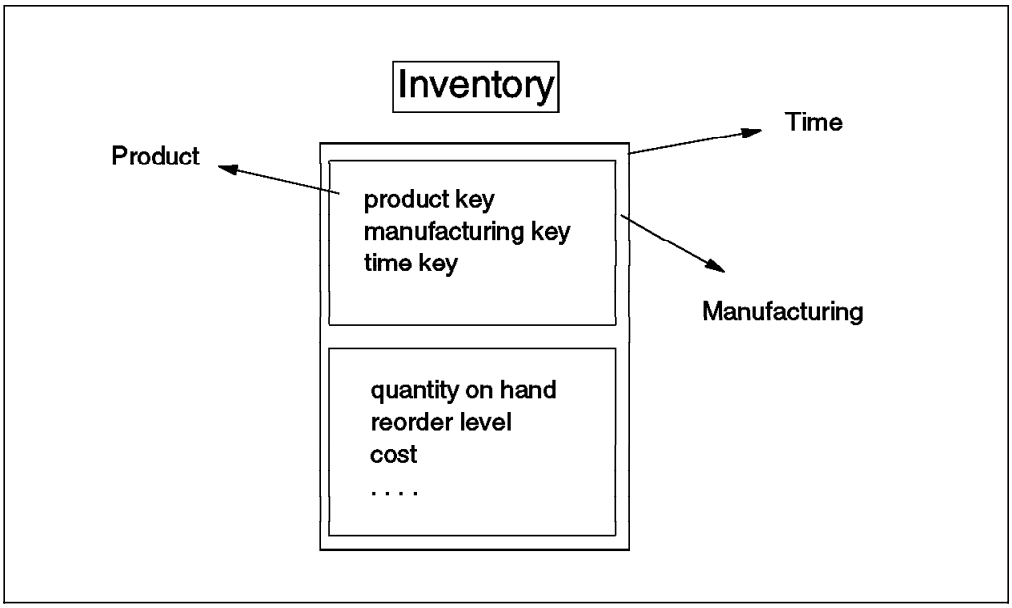


Figure 50. Inventory Fact Representing the Inventory State.

A third solution to this problem consists of changing the semantics of the Inventory fact, from representing the Inventory state to Inventory state changes.

Facts Representing Changes to the State of Business Objects. If we further investigate the representativity of Average Quantity On Hand for the Inventory business object, increasing the time dimension granularity from, say, a month to a day, we may still have a fundamental problem if Quantity on Hand really changes frequently. By increasing the granularity of the time dimension for state-related facts, we can assume that the representativity problem of the measures becomes less severe but may not disappear entirely (see Figure 51 on page 104).

If we want to provide information analysts with a solution for fine-grained analysis of the behavior of business objects like Inventory, we have to change the semantic interpretation of the fact. For the Inventory Fact, for instance, we have to capture the Inventory state changes in our dimensional model.

In reality, it usually is difficult to decide whether a state model or a state change model for a business object is to be preferred. Some users may have to work predominantly with states, others with state changes, even for facts related to the same business objects. The essence of the problem is one of time-variancy modeling, and we deal with this in much more detail in “Modeling Time-Variancy of the Dimension Hierarchy” on page 137.

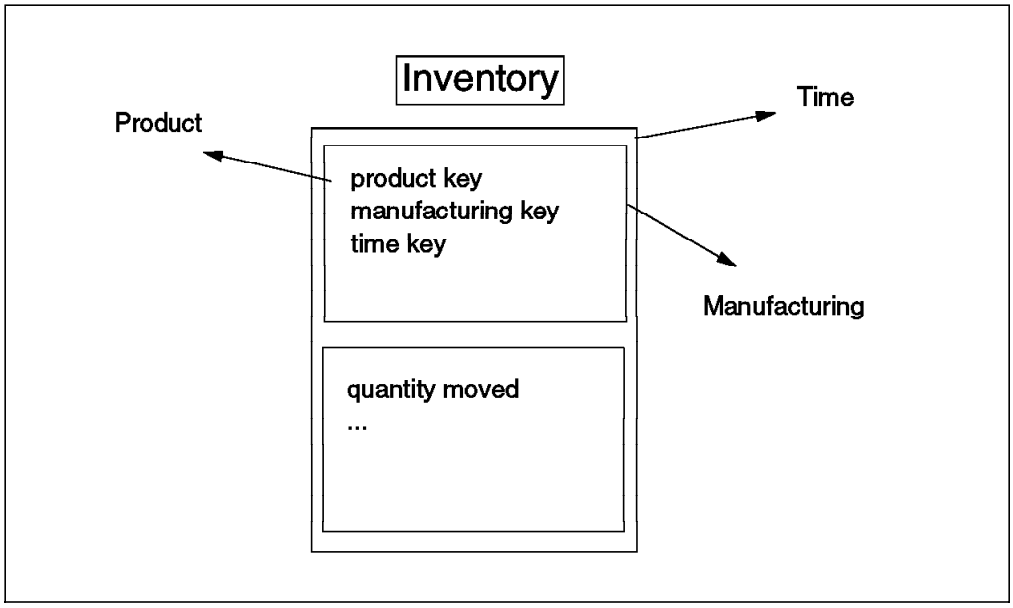


Figure 51. Inventory Fact Representing the Inventory State Changes.

To conclude, make sure you are aware of the fundamental differences between both solutions for modeling facts related to business objects. The Inventory state model and the Inventory state change model in both solutions may carry the same names (although we do not recommend doing this), but the facts they represent clearly are totally different: one says how much we had in stock for a certain product during a given period of time, the other would say how the stock changed, for a certain product, over time. Also, the time dimension for both is fundamentally different. In the state model, the time dimension must be interpreted as a duration. In the state change model, the time dimension must be interpreted as a ″time stamp″ rather than a duration. It is clear that the users must be made fully aware of this, which stresses the importance of business metadata.

Business-Related Requirements Analysis. In the previous sections, we have seen several examples of situations where requirements analysis and initial dimensional modeling done from a business-related perspective result in better solutions than if the work is done strictly from the analysis of available end-user requirements.

Requirements analysis based on captured end-user requirements should be considered as an aid in the process. If well done, it will lead to a solution that addresses the end users′ perceptions of the work they do and consequently works in practice. You must expect, however, that such solutions will have a narrow scope of coverage of the business problem domain and therefore not last very long. As a general guideline, we recommend performing business-related requirements analysis and initial dimensional modeling.

Because of the straightforward semantics which can be associated with measures, dimensions, and facts in a dimensional model, some dimensional modelers prefer to identify facts before anything else in the model. They look for ″things″ (business objects, transactions, and events) that are of fundamental importance for the business process they want to analyze. This ″gives″ them the candidate facts. Likewise, identifying the elements that identify the facts provides them with the candidate dimensions. Candidate measures can be identified based a study of what has to be tracked, sized, evaluated, etc. about these facts. If the general business context is well understood, business-related requirements analysis results in the creation of very representative initial dimensional models.

8.4.2.2 Creating the Initial Dimensional Model

With the candidate measures, dimensions, and facts, an initial dimensional model can be built. Figure 52 presents two such initial models, one for the Sales fact and one for the Inventory fact for the CelDial case study.

Experience has shown that most information analysts can fully understand these schemas, even though they represent structured dimension hierarchies in the model. We use such schemas for representing the initial model and discuss the potential usage of measures present in the model along the dimension paths, directly with the end users. In addition, these initial dimensional models are also sufficiently detailed for the modeling expert who subsequently has to develop the fully detailed dimensional models for the problem domain at hand.

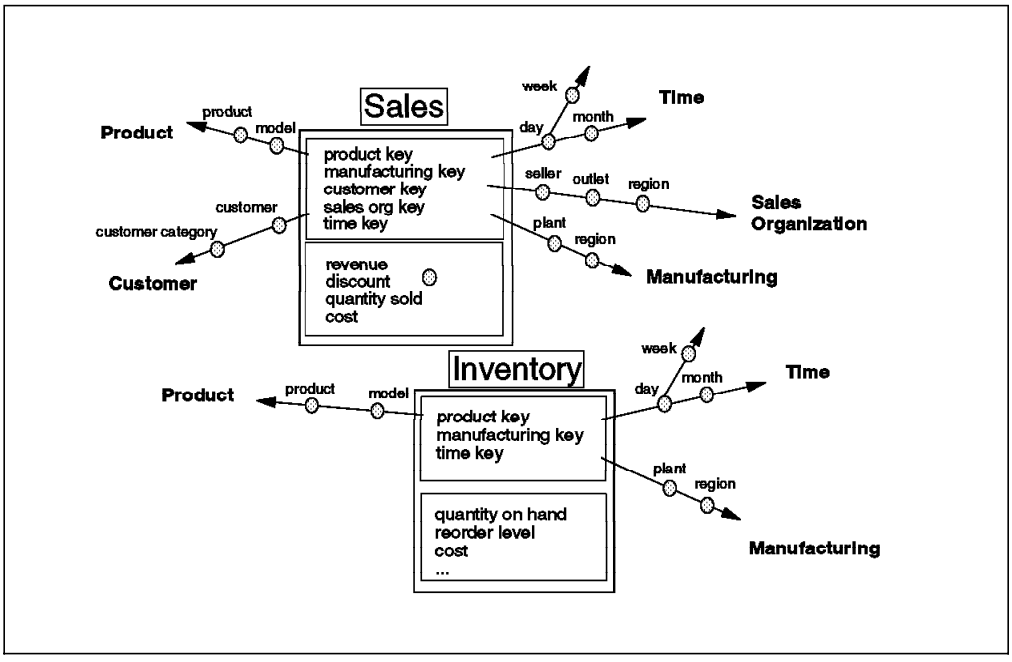


Figure 52. Initial Dimensional Models for Sales and Inventory.

Establishing the Business Directory: As part of the process of constructing the initial dimensional model, the base elements that make up a model and are directly related to the end-user's information analysis activities must be defined and described in what we call the *business* directory. The elements of the dimensional model are indeed core information items and their business definition should be established as precisely as possible.

We recommend that the business directory for the (new) elements of the dimensional model be created while the model is constructed because it is during the initial model construction that the elements are determined from among the set of end-user requirements. Unclear assumptions made about the meaning of candidate elements can put requirements analysis on the wrong track. Spelling out clear and precise definitions of the base elements will help produce an initial model that better represents the fundamentals of the business problem domain.

The base elements of the model must be defined in business terms. Each of the items is given a name and a definition statement of what it really represents. End users are actively involved in this process. They can either help write the definitions or validate them.

We recommend that this activity be performed rigorously. It is not a simple task to write precise definitions. Perhaps existing business dictionaries can help, if their definitions are up to date. Make sure, however, that the meaning of the modeling element is captured in these definitions, as the end users would define them.

These business directory definitions are the prime business metadata elements associated with the initial dimensional model. They will become part of the business metadata dictionary for the data warehouse. End users will use these definitions in their information analysis activities, to explore what is available in the data warehouse and interpret the results of their information analysis activities.

Determining Facts and Dimension Keys: One of the guidelines stated in "Candidate Facts" on page 100 says that facts should be uniquely identifiable merely by the existence of their real-life equivalent. Whether or not this implies that facts in fact tables should have a unique identifier is a debatable issue. Especially for transaction and event-related facts, it is not clear that nonunique facts in the fact table are really harmful.

Determining Facts. For facts that relate to states of business objects, the need to be able to uniquely identify the business object's state at a particular point in time is more of an issue. For such facts, a unique identifier of the objects and their state is required. Good modeling practice then suggests that this guideline should be applied to all facts in a dimensional model, whatever they represent.

There are several ways of identifying facts uniquely. The most straightforward way is through combining their base dimensions: The Inventory fact in the CelDial model is identifiable naturally through combining a product, a manufacturing plant, and a period of time (for instance a particular day or month). Notice that the presence of a time period or time duration in a fact's identification can lead to an awkward looking "key" when the time period is a less obvious period than day or month. This is for instance the case when time periods are used that are determined by a begin- and end-time, a technique for modeling state changes that can occur at any point in time and last for a period that is essentially of varying length. Because we can assume that in such cases no two facts related to the same business object could occur or should be registered with the same begin-time, we can use the begin-time as one of the elements to uniquely identify the facts.

Identifying facts using their base dimensions is interesting from another perspective. Analyzing the facts in a dimensional model from the perspective of

their identifying dimensions can further clarify issues about the granularity of facts. Two examples can illustrate this:

- **Example 1**: If the inventory state fact is identified through the product model identifier, in combination with an identifier for the manufacturing plant at which the inventory resides, it is clear that the model cannot provide support for investigating inventorization of product components (which are of a lower granularity than product models) and cannot be used to analyze inventories within manufacturing plants (there may be several inventory locations in each of the plants). Figure 53 shows an example of an inventory state fact with lower level granularities for the Product and Plant dimensions.

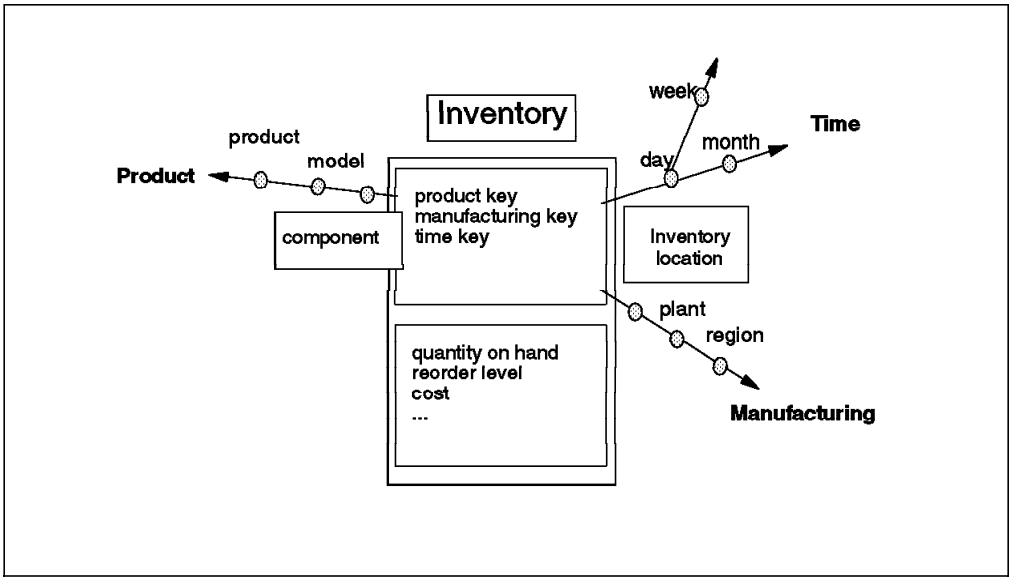


Figure 53. Inventory State Fact at Product Component and Inventory Location Granularity.

- **Example 2**: If an Inventory state change fact were used in the dimensional model and if the time dimension of the Inventory fact were used at the granularity of a day, there is no guarantee that all facts in the Inventory fact table would be unique: Several inventory changes can indeed occur, for a given Product Model in a given Plant, during a particular day. To solve this situation, the model could for instance add the Inventory Movement Transaction dimension key to the Inventory fact (see Figure 54 on page 108). This dimension key can have several different forms: It can be associated with a business document number, possibly combined with the location of the inventory where the move is to take place, or it can be a system or technical attribute that makes the fact unique (e.g., a microsecond time stamp that represents the time when the inventory change takes place, possibly combined with the inventory location).

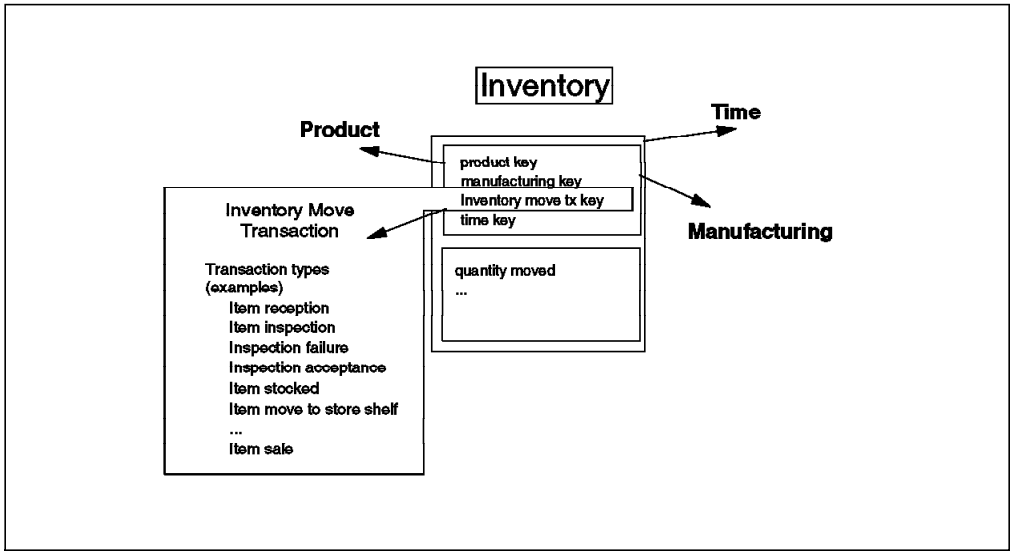


Figure 54. Inventory State Change Fact Made Unique through Adding the Inventory Movement Transaction Dimension Key.

Facts are identified through either system-generated keys or composite keys that consist of their dimension keys, at the lowest granularity level. For facts that represent transactions, events, or state changes, the system-generated key could be a transaction identifier or a fine-granular time stamp, possibly enhanced by adding the source data system identifier to it. For state-oriented facts, it usually is more difficult to find representative system-generated keys. It should also be pointed out that system-generated keys for facts can introduce difficulties for the populating subsystem. Although we cannot eliminate the technique completely, we do not recommend it, at least not when other approaches are viable.

To reduce the complexity of the solution, we highly recommend avoiding having composite dimension keys in the dimensional model.

Determinant Sets of Dimension Keys. Facts in a dimensional model can usually be identified through different combinations of dimension keys (see Figure 55 on page 109). This situation occurs quite often. If facts are analyzed by different groups of end users, each with a different perspective on the analysis problem domain, more dimension keys will be determined and more combinations of dimension keys will become possible.

Not all combinations of dimension keys present within a given fact are valid or even meaningful. Figure 55 on page 109 shows valid and invalid combinations of dimension keys. Each valid combination of dimension keys for a particular fact is called a determinant set of dimension keys. Each fact in the model can have several such determinant sets of dimension keys, and it is good modeling practice to identify these determinant sets clearly. The user should be informed, through the business metadata, which sets of determinant dimension keys are available for each fact.

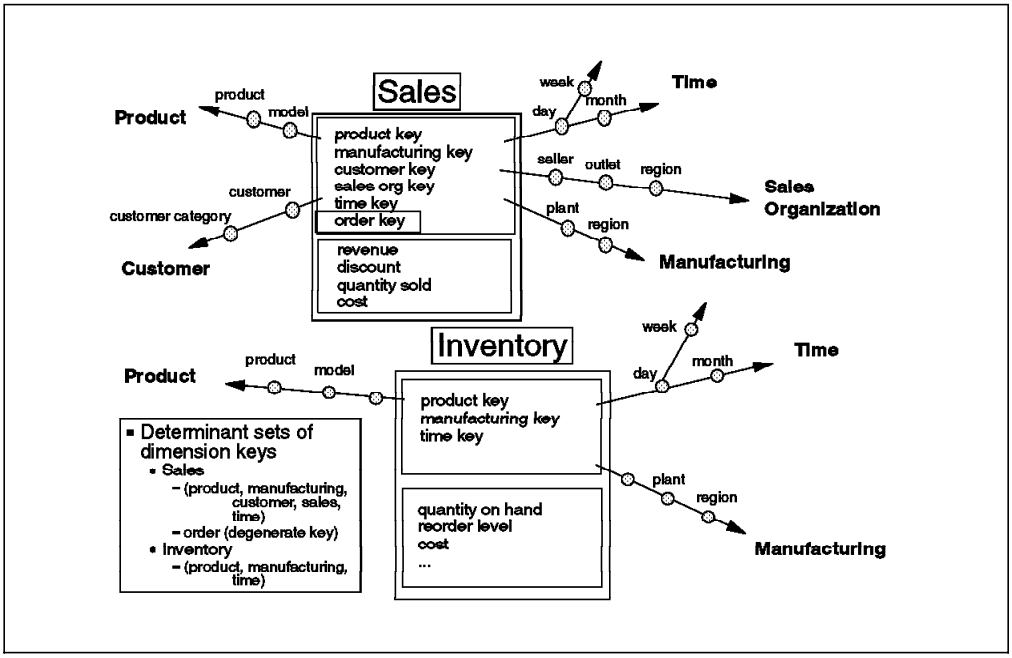


Figure 55. Determinant Sets of Dimension Keys for the Sales and Inventory Facts for the CelDial Case.

Determining Representative Dimensions and Detailed Versus Consolidated Facts: What are representative dimensions? To answer this question, we have to consider the question at two levels. The base level question is: What are representative dimensions for the particular end-user requirements we are considering right now? The second level question is: What are representative dimensions for facts, knowing that the model we have at a particular point in time will have to be integrated with other dimensional models, each considering possibly distinct sets of end-user requirements?

The issues involved in solving the base question can be illustrated by using the Sales fact in the CelDial case study. According to the business process description, we (may) have to differentiate between a Corporate Sale and a Retail Sale. Corporate Sales can either be handled by one of CelDial's SalesPersons, or the buying corporation can directly issue the order to an OrderDesk. Retail Sales fit less well into this Corporate Sales pattern, however. At first, we may think of having different (detailed or type-specific) facts in our model, one fact representing corporate Sales and the other Retail Sales. In this case, we would almost naturally consider two distinct dimensions, too: one for the Corporate Sales Organization, the other for the Retail Sales Organization (see Figure 56 on page 110).

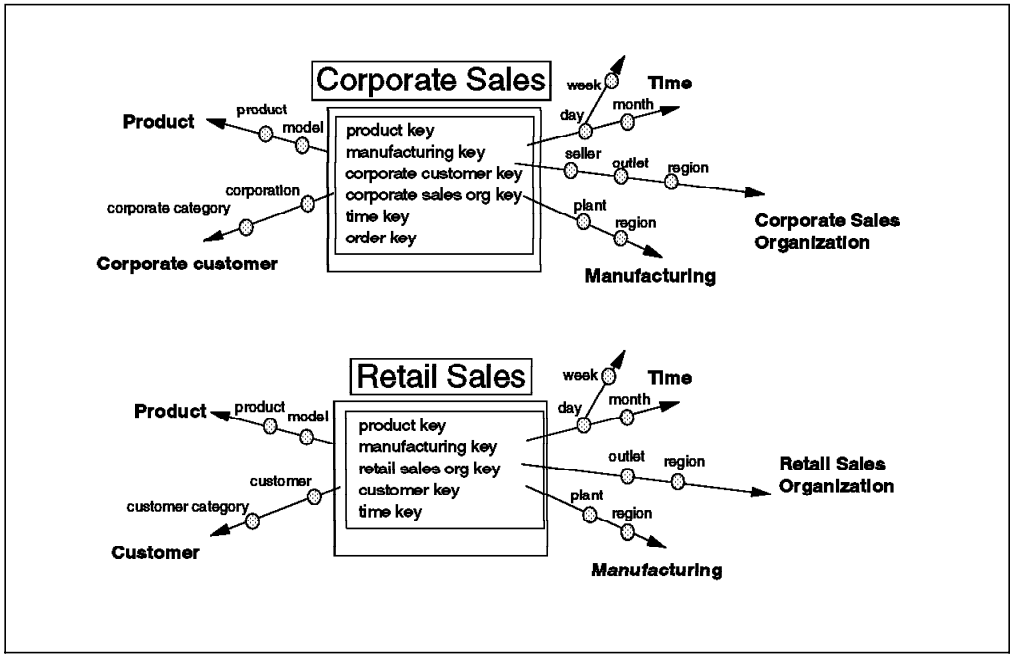


Figure 56. Corporate Sales and Retail Sales Facts and Their Associated Dimensions.

Separating Sales over two detailed facts has several important implications. One of the prime ones is that this approach does not fit well should someone in the CelDial organization want to make a consolidated analysis of Sales Revenue, disregarding the difference between corporate and retail sales. If this indeed happens frequently, it would be best to merge all Sales facts into a single consolidated fact table. But once we do that, what do we do with the two separate dimensions we have: Corporate Sales Organization and Retail Sales Organization.

There are basically two solutions we could apply (see Figure 57 on page 111): either we keep the dimensions separate or we merge them into one single dimension called Sales Organization. In the first case, we have to have a type indicator in the Sales fact table with which we can determine whether we are dealing with a Corporate Sale or a Retail Sale. This type indicator is required because we have to be able to join facts with the correct dimension. In the second case, we do not need this indicator, but we must make sure that both dimensions can be happily merged into a single dimension. Determining whether a Sale is a Corporate or a Retail Sale should be done from information we can find in the Sales Organization dimension itself. The indicator is not needed for the join, in this case.

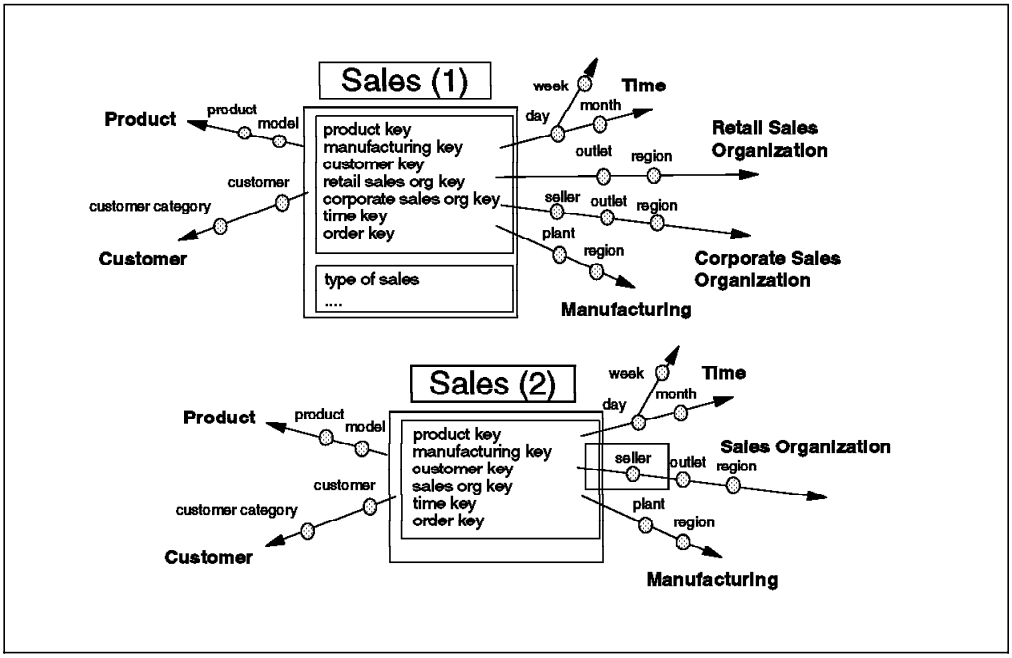


Figure 57. Two Solutions for the Consolidated Sales Fact and How the Dimensions Can Be Modeled.

In this particular case, the first alternative is the preferred one. It simplifies the model, and it supports global sales revenue analysis as well as detailed analysis of corporate sales and retail sales. The choice of solution is obvious in this case because both dimensions (Corporate Sales Organization and Retail Sales Organization) have so much in common for CelDial's business organization. When investigating modeling of the dimensions in a dimensional model in more detail, we will see which criteria can be used to assess whether dimensions can be and should be merged or not and how this could have an impact on consolidating facts.

Determining a representative set of corporate dimensions is a very important aspect of dimensional modeling. Integrating several individually developed dimensional models in a global, shared data mart that not only supports fact analysis of "isolated" groups of end users but also allows for fact-fact analysis depends on the presence in the model of common dimensions. For readers who are familiar with the basics of the relational model, this should not be a big surprise: if you want to join one fact with another, you need a join attribute in both facts that is drawn from the same domain. In dimensional modeling, this can be achieved only if facts can be associated with perfectly identical dimensions.

Dimensions and Their Roles in a Dimensional Model: There is yet another important aspect related to the choice of dimension keys in facts and dimensions in a dimensional model. Consider for instance the situation where the Sales fact has several time dimension keys: Order date, Shipment date, Delivery date, etc. All of these time dimension keys relate the fact to apparently the same time dimension. However, this time dimension acts in different roles with respect to the Sales fact. A similar situation could occur with any other non-time-related dimension key in a dimensional model

Dimensions that appear several times in different roles are a very common situation in dimensional models. Although the situation can easily be solved, we

do have to take it into account in our modeling approach. Dimension keys in fact tables should be given names that reflect the roles they play for the fact. A dimension key called *Time* is therefore not a very good idea. From the examples presented above, we should provide names for the various time dimensions such as Order Date, Shipment Date, and Delivery Date (see Figure 58).

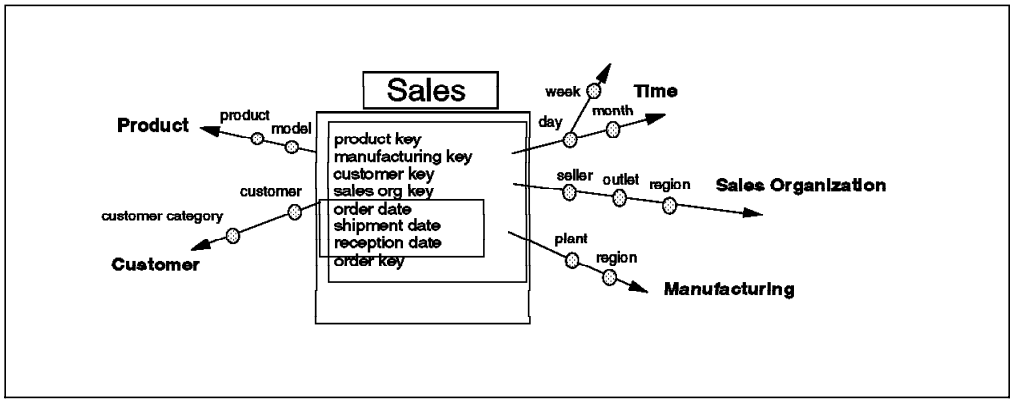


Figure 58. Dimension Keys and Their Roles for Facts in Dimensional Models.

Getting the Measures Right: Measures are elements of prime importance for a dimensional model. During the initial dimensional modeling phase, candidate measures are determined based on the end-user queries and their requirements in general. Candidate measures identified in this way may not be the best possible choices. We strongly suggest that each and every candidate measure be submitted to a detailed assessment of its representativity and its usefulness for information analysis purposes.

It is generally recommended that the measures within the dimensional model be representative from a generic business perspective. Failing to do so will make models nonintuitive and complicated to handle. Failing to do so also will make the dimensional model unstable and difficult to extend beyond a pure local interest.

When investigating the ″quality″ of candidate measures, you should focus on the following main issues:

- **Meaning of each candidate measure:** Expressed in business terms, a clear and precise statement of what the measure actually represents is a vital piece of metadata that must be made available to end users.
- **Granularities of the dimensions of each measure:** Although granularities of dimensions are usually considered at the level of facts, it is important that measures incorporated within a fact are evaluated against the dimension keys of that fact. Such an evaluation may reveal that a given measure may better be incorporated in another fact or that granularities should perhaps be changed. Particular attention should be paid to analyzing the meaningfulness of the candidate measures versus the time dimension of the fact.
- **Relationship between each measure and the source data item or items it is derived from:** Although there is no guarantee that source data items are actually correct representations of business-related items, it is clear that measures are derived from these source data items and that therefore this derivation must be identified as clearly and precisely as possible. You may have to deal with very simple derivations such as when a measure is an import of a particular source data item. You may also have to deal with complex derivation formulas, involving several source data items, functional

transformations such as sums, averages or even complex statistical functions, and many more. This information is of similar if not more importance than the definition statement that describes the meaning of the measure in business terms. Unfortunately, this work is seldom done in a precise way. It is a complex task, especially if complex formulas are involved. The work usually is further complicated because of replication and duplication of data items in the source data systems and the lack of a source data business directory and a precise understanding of the data items in these systems. Nevertheless, we strongly advocate that this definition work be done as precisely as possible and that the information is made available to end users as part of the metadata.

- **Use of each measure in the data analysis processes:** Measures are used by end users in calculations that are essential for producing ″meaningful″ data analysis results. Calculations such as these can be simple such as in these examples:
  1. Display a list of values of a particular measure, for a selection of facts. Other calculations can involve complicated formulas.
  2. Assuming products shipped to customers are packaged in cases or packaging units, to calculate the Quantity Shipped of a given product in an analytical operation that compares these numbers, for products that can be packed in different quantities, the formula should in some way include packaging conversion rules and values. These calculations may involve a sequence of related calculations.
  3. To calculate the Net Profit of a Sale, we may first have to calculate several kinds of costs and a Net Invoice Price for the Sale before calculating the Net Profit.

For a data warehouse modeler, it is essential to capture the fundamental calculations that are part of the information analysis process and assess their impact on the dimensional model. Two fundamental questions must be investigated each time: Can the calculation be performed? In other words, does the model include all of the data items required for the calculation? And, Can the calculation be performed efficiently? In this case you are assessing primarily how easy it is for the analyst to formulate the calculation. If feasible, some performance aspects associated with the calculations may be assessed here too.

In practice, it is clear that analyzing each and every calculation involving a particular measure or a set of measures is impossible to do. What we suggest, however, is that the modeler take the time to analyze the key derivation formulas of the data analysis processes. The purpose is to find out whether the candidate measures are correctly defined and incorporated in the model. This work is obviously heavily influenced by the available end-user requirements, knowledge of the business process, and the analytical processing that is performed.

In addition to evaluating the key derivation formulas and how they impact the dimensional model, building a prototype for the dimensional model is a very welcome aid for this part of the work. As with any ″learning″ process, the prototype may be filled up with a sampling of source data and made available to end users as a ″training set.″

Measures are also heavily involved in the typical OLAP operations: slicing, rollup, drilldown. Here too, some assessment of the measures involved in these operations may help improve the model. The ″quality″ analysis of the measures for these cases is somewhat simpler than the above, though. In fact, a dominant

question related to all these operations is whether a particular measure is additive or not, and whether this property is applicable to all of the dimension keys of the measure or only to some.

Ralph Kimball defines three types of measures (Ralph Kimball, *The Data Warehouse Toolkit*):

- **Additive**: Additive measures can be added *across* any of their dimensions. They are the most frequently occurring measures. Examples of additive measures in the CelDial model are: Total Cost and Total Revenue.
- **Semiadditive**: Semiadditive measures can be added only across some of their dimensions. An example in the CelDial model is Average Quantity On Hand in the Inventory fact, which is not additive across its time dimension.
- **Nonadditive**: Nonadditive measures cannot be added across any of their dimensions. Frequently occurring examples of nonadditive dimensions in dimensional models are ratios.

Semiadditive and nonadditive measures should be modeled differently to make them (more) additive: otherwise, the end user must be made aware of the restrictions.

Fact Attributes Other Than Dimension Keys and Measures: So far, our fact tables have only contained dimension keys and measures. In reality, fact tables can contain other attributes too. Because fact tables tend to become very large in terms of the number of facts they contain, we recommend being very selective when adding attributes. Very specifically, all kinds of descriptive attributes and labels should be avoided within facts. In reality though, adding one or more attributes to a base fact can make querying much more easy without causing too much of an impact on the size of the fact and consequently on the size of the fact table itself.

Several of the attributes the modeler will want to add to a fact will be derived attributes. For a data warehouse model, adding derived attributes should not really be a problem, particularly because the data warehouse is a read-only environment. When adding derived attributes to a fact, however, the modeler should understand and assess the impact of adding attributes on the data warehouse populating subsystem. Usually, adding derived attributes anywhere in the data warehouse model is a trade-off between making querying easier and more efficient and the populating process more complicated.

Three types of fact attributes are particularly interesting to consider. They are illustrated with the CelDial model in Figure 59 on page 115.

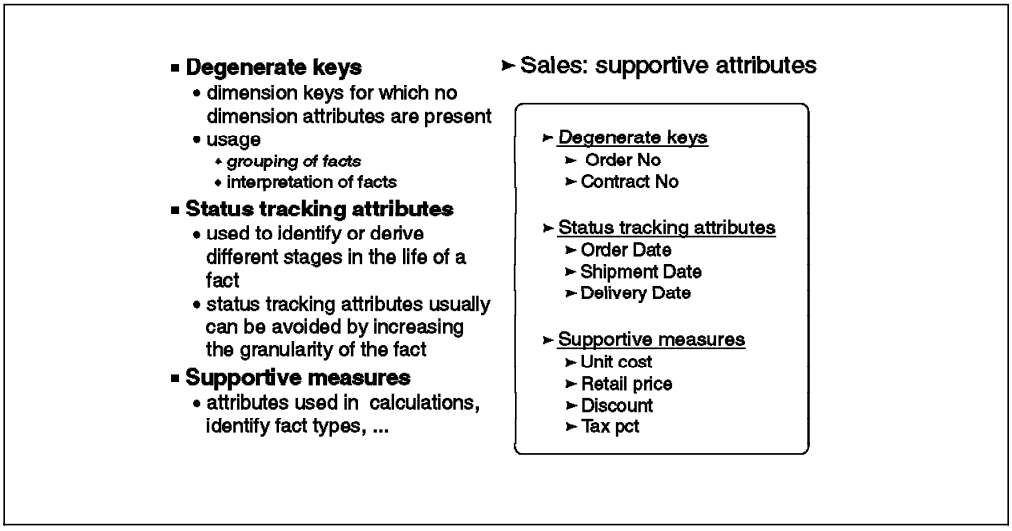


Figure 59. Degenerate Keys, Status Tracking Attributes, and Supportive Attributes in the CelDial Model.

Degenerate keys are equivalent to dimension keys of a fact, with the exception that there is no other dimension information associated with a degenerate key. Degenerate keys are used in data analysis processes to group facts together: for example, in the CelDial model, SalesOrder is represented through the Order dimension key in the Sales fact.

Status tracking attributes identify different states in which the fact can be found. Often, status tracking attributes are status indicators or date/time combinations. Status tracking attributes are used by the information analyst to select or classify relevant facts. Their appearance in a fact table often is related to the granularity of the dimensions associated with the fact. For example, in the CelDial model, the Sales fact may contain status tracking attributes that indicate whether the Sale is ″Received,″ ″In process,″ or ″Shipped.″ This can either be modeled using a state attribute or three date/time attributes representing when the sale was received, being processed or shipped to the customer.

Supportive attributes are added to a fact to make querying more effective. Supportive attributes are those a modeler has to be particularly careful with, because there is often no limit to what can be considered as being supportive. For example, in the CelDial model, Unit Cost in the Sales fact could be considered a supportive attribute. Other frequently occurring examples of supportive attributes are key references to other parts in the dimensional model. These attributes help reduce complex join operations, which end users should otherwise have to formulate.

8.4.3 Requirements Validation

During requirements validation, the results of requirements analysis are assessed and validated against the initially captured end-user requirements. Also as part of requirements validation, candidate data sources on which the end-user requirements will have to be mapped are identified and inventoried. Figure 60 on page 116 illustrates the kinds of activities that are part of requirements validation.

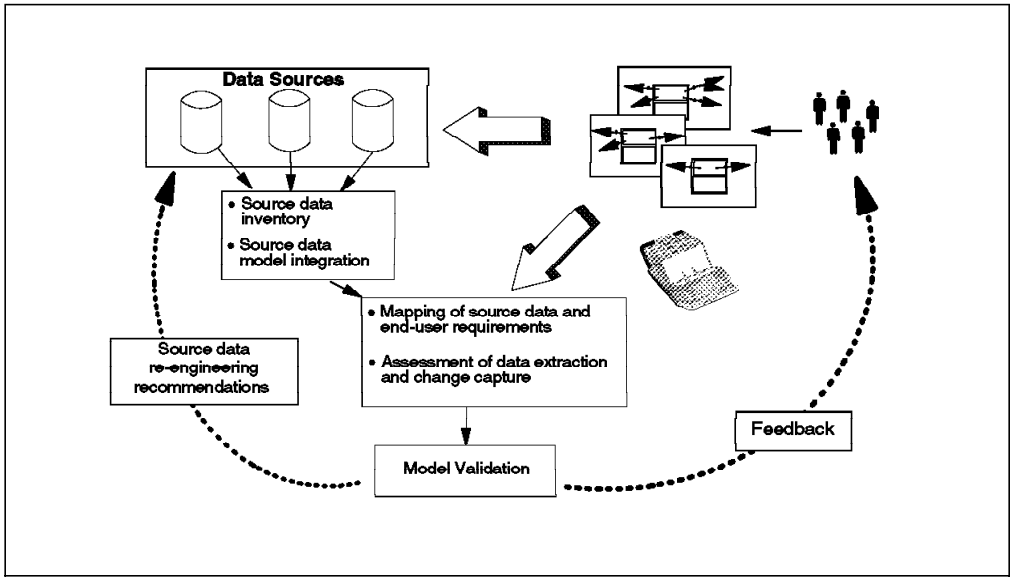


Figure 60. Requirements Validation Process.

The main activities that have to be performed as part of requirements validation are:

- Checking of the coherence and completeness of the initial dimensional models and validation against the given end-user requirements. The initial models are analyzed with the end users. As a result, more investigations could be performed by the requirements analyst and the initial models may be adapted, in an attempt to fix the requirements as they are expressed in the models, before passing them to the requirements modeling phase.
- Candidate data sources are identified. An inventory of required and available data sources is established.
- The initial dimensional models, possibly completed with informal end-user requirements, are mapped to the identified data sources. This is usually a tedious task. The source data mapping must investigate the following mapping issues:
  - Which source data items are available and which are not? For those that are not available, should the source applications be extended, can they perhaps be found using external data sources, or should end users be informed about their unavailability and as a consequence, should the coverage of the dimensional model be reduced?
  - Are other interesting data items available in the data sources but have not been requested? Identifying data items that are available but not requested may reveal interesting other facets of the information analysis activities and may therefore have significant impact on the content and structure of the dimensional model being constructed.
  - How redundant are the available data sources? Usually, data items are replicated several times in operational databases. Basically, this is the result of an application-oriented database development approach that almost automatically leads to disparate operational data sources in which lots of data is redundantly copied. Studying redundant sources involves studying data ownership. This study must identify the prime copy of the source data items required for the dimensional model.

- Even if the source data items are available, one still has to investigate whether they can be captured or extracted from the source applications and at what cost. As part of requirements validation, a high-level assessment of the feasibility of source data capturing must be done. Feasibility of data capture is very much influenced by the temporal aspects of the dimensional model and by the base granularities of facts and measures in the model.
- To conclude the requirements validation phase, an initial sizing of the model must be performed. If possible at all, the initial sizing should also investigate volume and performance aspects related to populating the data warehouse.

The results of requirements validation must be used to assess the scope and complexity of the data warehouse development project and to (re-)assess the business justification of it. Requirements validation must be performed in collaboration with the end users. Incompleteness or incorrectness of the initial models should be revealed and corrected. Requirements validation may involve building a prototype of the dimensional model.

As a result of requirements validation, end-user requirements and end-user expectations should be confirmed or reestablished. Also as a result of requirements validation, source data reengineering recommendations may be identified and evaluated. At the end of requirements validation, a (new) ″sign-off″ for the data warehouse modeling project should be obtained.

8.4.4 Requirements Modeling - CelDial Case Study Example

Requirements modeling consists of several important activities that all are performed with the intent of producing a detailed conceptual model that represents at best the problem domain of the information analyst. Figure 61 gives an overview of the major activities that are part of requirements modeling. Obviously, the project itself determines to what extent each of these activities should be performed.

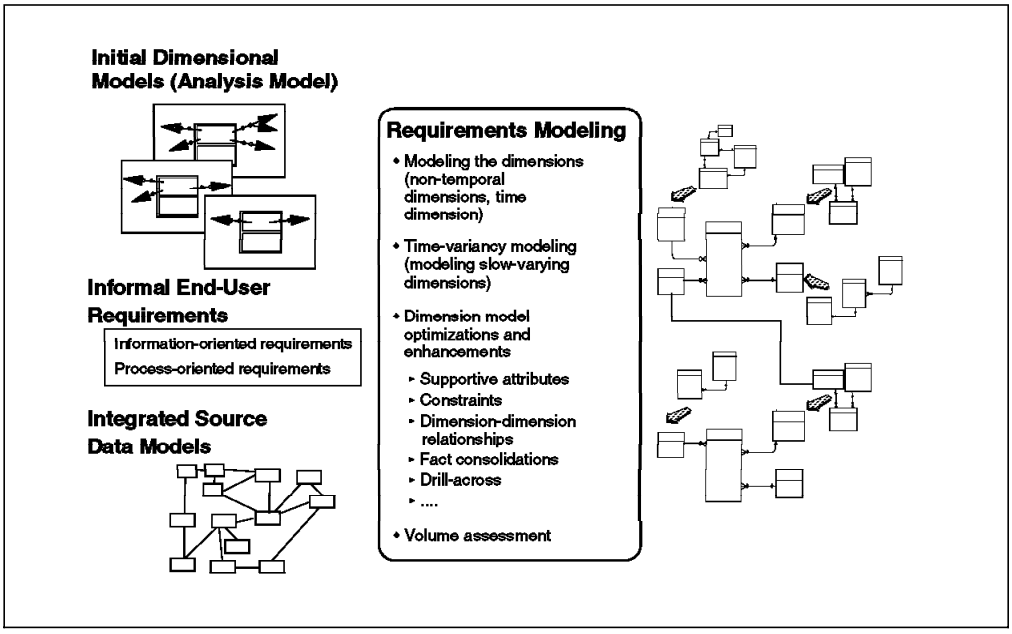


Figure 61. Requirements Modeling Activities.

Modeling the dimensions consists of a series of activities that produce detailed models for the various candidate dimensions which are part of the initial dimensional model.

A detailed dimension model should incorporate all there is to capture about the structure of the dimension as well as all of its attributes. One approach consists of producing the dimension models in the form of a flat dimension table. This approach results in models called star models or star schemas. Another approach produces dimension models in the form of structured ER models. This approach is said to produce so-called snowflake models or snowflake schemas. Figure 62 illustrates the star model approach and Figure 63 illustrates the snowflake approach for the Celdial case study.

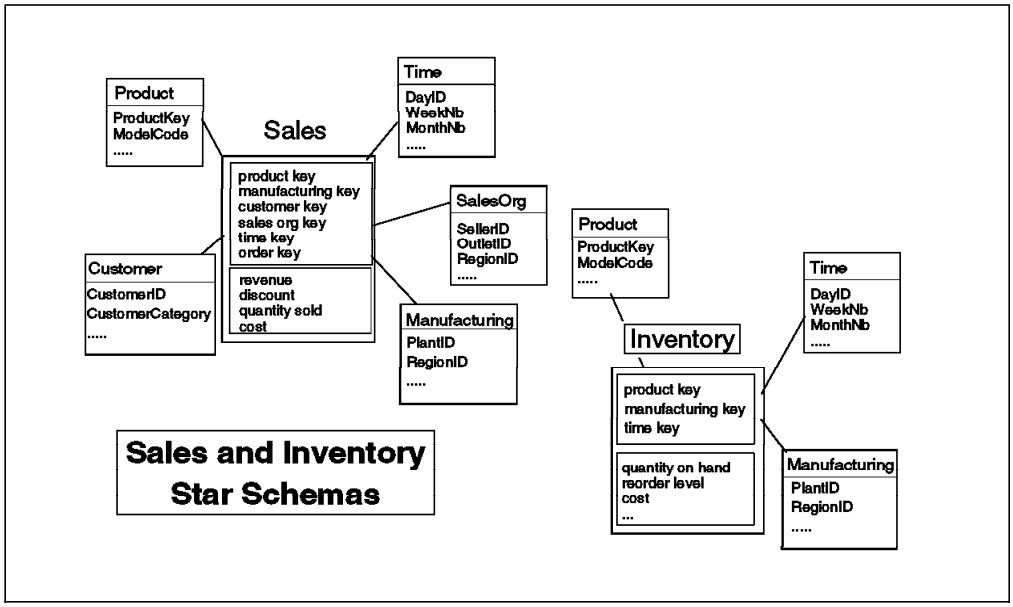


Figure 62. Star Model for the Sales and Inventory Facts in the CelDial Case Study.

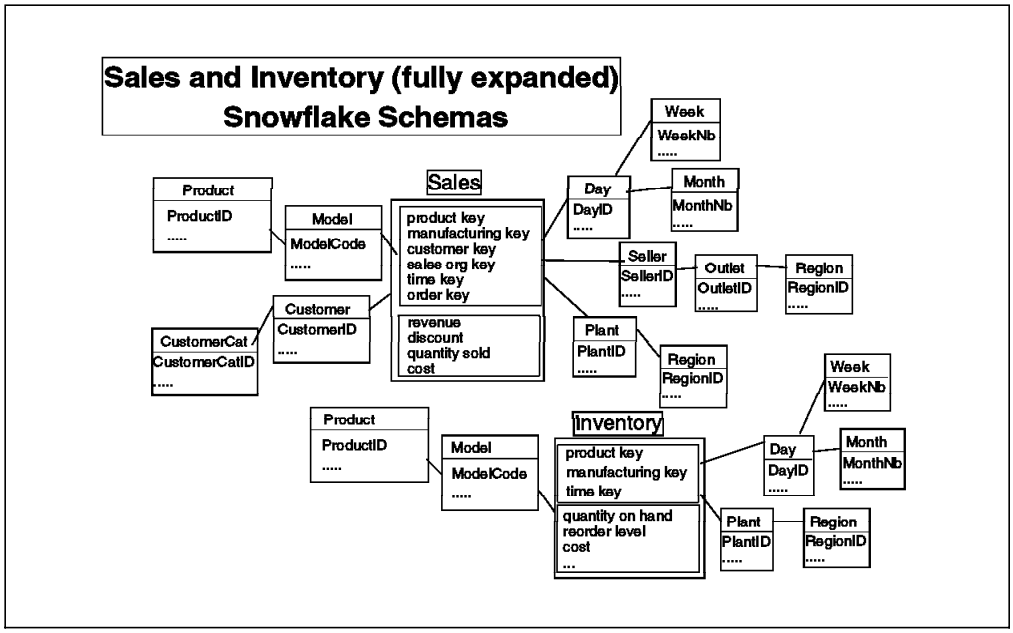


Figure 63. Snowflake Model for the Sales and Inventory Facts in the CelDial Case Study.

Dimensions play a particular role in a dimensional model. Other than facts, whose primary use is in calculations, the dimensions are used primarily for:

1. Selecting relevant facts
2. Aggregating measures

The base structure of a dimension is the hierarchy. Dimension hierarchies are used to aggregate business measures, like Total Revenue of Sales, at a lesser level of detail than the base granularity at which the measures are present in the dimensional model. In this case, the operation is known as roll-up processing. Roll-up processing is performed against base facts or measures in a dimensional model.

To illustrate roll up: Sales Revenue at the Regional level of CelDial's Sales Organization can be derived from the base values of the Revenue measure that are recorded in the Sales facts, by calculating the total of Sales Revenue for each of the levels of the hierarchy in the Sales Organization.

If measures are rolled up to a lesser level of detail as in the above example, the end user can obviously also perform the inverse operation (drill down), which consists of looking at more detailed measures or, to put it differently, exploring the aggregated measures at lower levels of detail along the dimension hierarchies. Figure 64 illustrates roll-up and drill-down activities performed against the Inventory fact in the CelDial case.

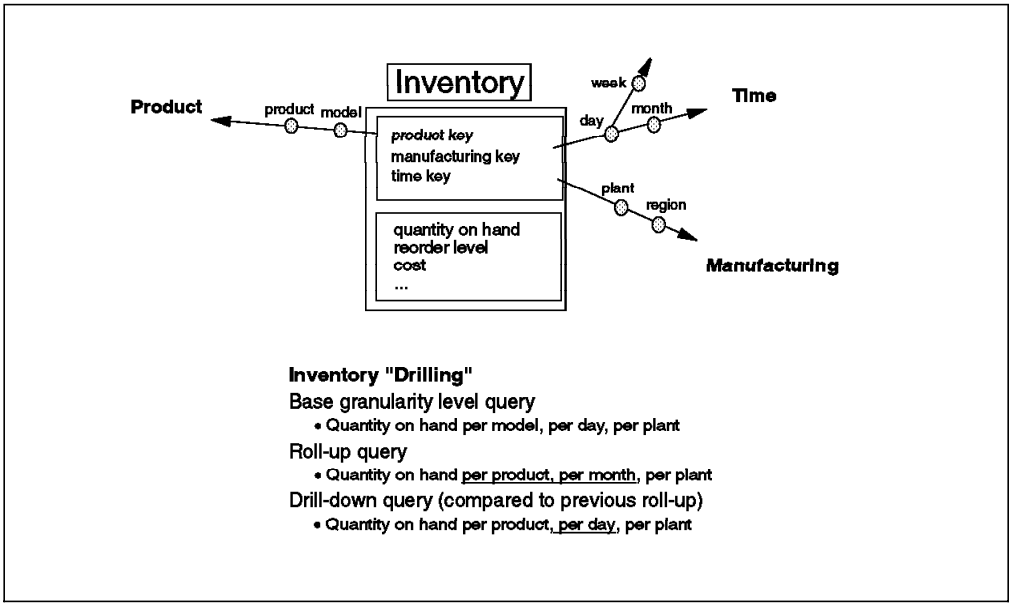


Figure 64. Roll Up and Drill Down against the Inventory Fact.

For all of the above reasons, dimensions are also called *aggregation paths* or *aggregation hierarchies*. In real life, where pure hierarchies are not so common, a modeler very frequently has to deal with dimensions that incorporate several different parallel aggregation paths, as in the example in Figure 65 on page 120.

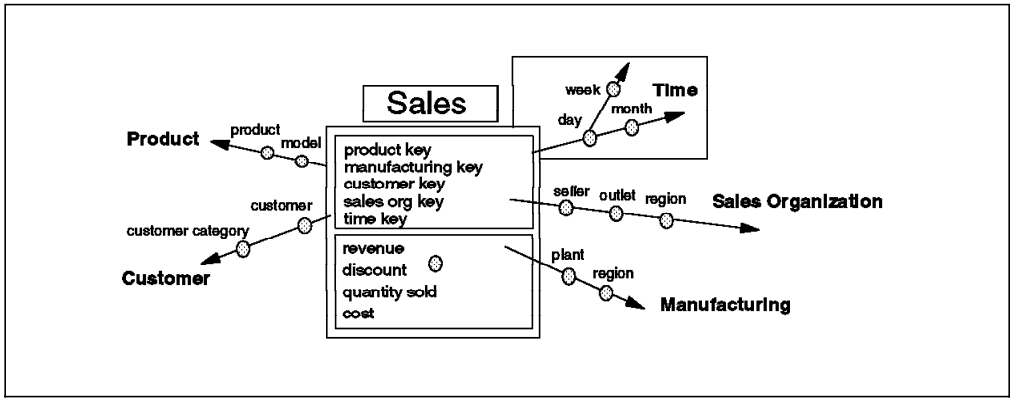


Figure 65. Sample CelDial Dimension with Parallel Aggregation Paths.

One of the essential activities of dimension modeling consists of capturing the aggregation paths along which end users perform roll up and drill down. The models of the dimensions produced as the result of these activities will further be extended and changed when other modeling activities are performed, such as modeling the variancy of slow-varying time dimensions, dealing with constraints within the dimensions, and capturing relationships and constraints across dimensions. These elaborated modeling activities can have an impact on the dimensional model as a whole.

Now, let us explore the basics of dimension modeling (notice the subtle textual difference between dimension modeling and dimensional modeling....), developing models for some representative nontemporal dimensions for CelDial, as well as for the time dimension.

8.4.4.1 Modeling of Nontemporal Dimensions

Figure 66 illustrates the Sales and Inventory facts in the CelDial case study with their associated dimensions: Product, Manufacturing, Customer, Sales Organization, and Time. Let′s explore the representative nontemporal dimensions in the CelDial case study.

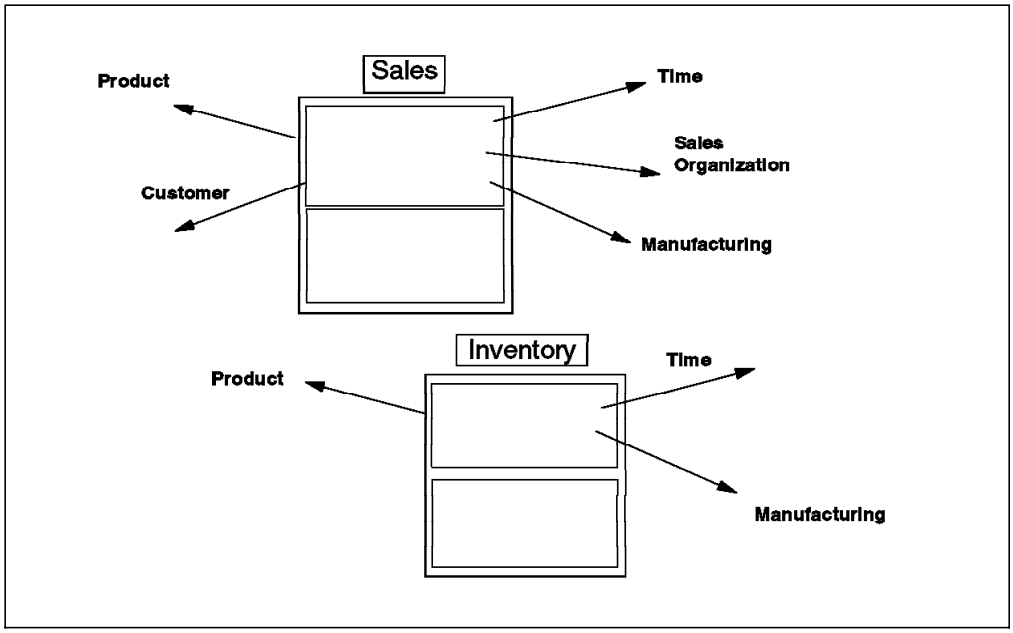


Figure 66. Inventory and Sales Facts and Their Dimensions in the CelDial Case Study.

Notice that the dimensions in CelDial's models in Figure 65 are extremely simple. The Manufacturing dimension, for instance, consists of a manufacturing key, a region, and a plant name. This provides support for selecting facts associated with given manufacturing units or plants and aggregates them at regional level. The Product dimension has some more properties, but still it is only partly representative of reality: because we have captured particular end-user requirements, we should expect to find only part of what the real model should incorporate. The simplicity of the model is a consequence of end-user focused development. Even though this approach may lead to an acceptable solution for the identified end users and the queries they expressed, it usually needs considerably more attention to produce a model that has the potential to become acceptable for a broad set of users in the organization. Failing to extend the solution model and in particular to make dimension models representative for a broad scope of interest results in stovepipe solutions where each group of end users has its own little data mart with which it is satisfied (for a while). Such solutions are costly to maintain, do not provide consistency beyond the narrow view of a particular group of users, and, as a consequence, usually lack integration capabilities. Such solutions should be avoided at all costs.

As a consequence, it is recommended that you consider modeling the dimensions in a broader context. We illustrate this next. The effects of this global approach to modeling the dimensions will become clear when we progress through our examples.

The Product Dimension: The Product dimension is one of the dominant dimensions in most dimensional models. It incorporates the complete set of items an organization makes, buys, and sells. It also incorporates all of the important properties of the items and the relationships among the items, as these are used by end users when selecting appropriate facts and measures and exploring and aggregating them against several aggregation paths that the product dimension provides.

CelDial's product context is inherently simple. Products are manufactured and models of the products are stocked in inventories in manufacturing plants, waiting for customers to buy them. In addition, end users are interested primarily in sales analysis and do not seem to attach a lot of importance to being able to analyze sales figures at different levels of aggregation. Product level and Regional level analysis seems to be what they want. In this situation, the product dimension built in the data warehouse can easily be represented by a flat structure, such as the Product dimension table in Figure 66 on page 120.

In most cases, however, the product dimension is a rather big component of the warehouse, potentially comprising several tens of thousands of items. The product dimension in a data warehouse usually is derived from the product master database, which is in most cases present in the operational inventory management system. You also have to consider that users usually show interest in far more extensive classification levels and classification types and that they handle potentially hundreds of properties of the items. It then should become clear that we should look at a broader context to bring out the real issues involved in dimension modeling for the Product dimension.

Let us therefore have a look at what could happen with the Product dimension, if CelDial were part of a large sales organization, comprising retail sales (mostly anonymous sales) as well as corporate sales. Figure 67 on page 122 provides

an initial model for the Inventory fact for our presumed extension of the primitive CelDial case. To study the problem, we use the initial dimensional modeling template introduced previously. Let us focus right here on the Product dimension only.

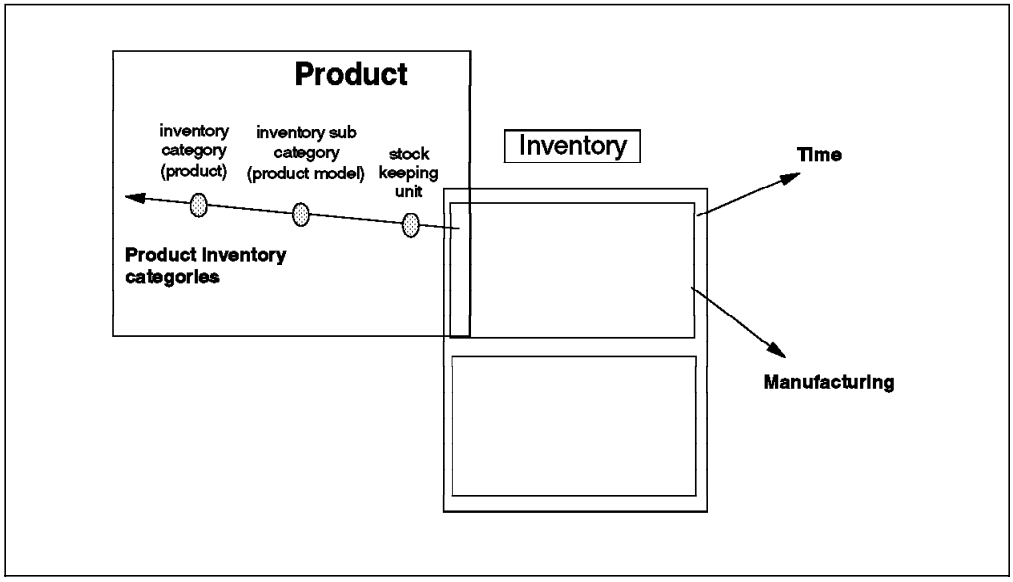


Figure 67. Inventory Fact and Associated Dimensions in the Extended CelDial Case Study.

The first observation to make is that although products are considered in sales operations, customers buy product models and not products. This implies that ″Product″ can in fact be considered a category of product models. In addition, if we also make the assumption that product models can be sold individually as well as packaged in units, the modeler must rearrange the Inventory fact and the Product dimension to account for that. In the situation illustrated in Figure 67, the model is rearranged as follows: We recognize that we need a concept called ″Stock Keeping Unit,″ which represents how a particular model of a product is available for sales to customers (be they anonymous or corporate customers) and how it is being sold to customers.

Consider there examples:

- A pack of 4 Duracell AA-batteries is a stock keeping unit. It consists of a particular production variant of Duracell′s AA-batteries (the product model). The product in this case is Duracell AA-batteries.
- Another stock keeping unit may be a particular cellular phone or some other electronic equipment, such as a notebook computer. These items definitely have variants in the form of product models, but one can assume they will be available for sales per unit. In this case, the stock keeping unit coincides with the product model.

  We recognize that a hierarchy of concepts exists in the problem domain and that users in fact work with these concepts frequently in their queries: the hierarchy is called the *Product Inventory Category* and it consists of three layers: Stock keeping unit (the base or the most granular layer), Inventory subcategory (which in our case coincides with product model), and Inventory category (equivalent to Product).

The difference with the CelDial model for Inventory in Figure 66 on page 120 is not yet very big, so let us consider one further step. Figure 68 on page 123

highlights the product dimension in the Sales fact in our extended CelDial case. Notice that we have incorporated the notion of stock keeping unit as described previously.

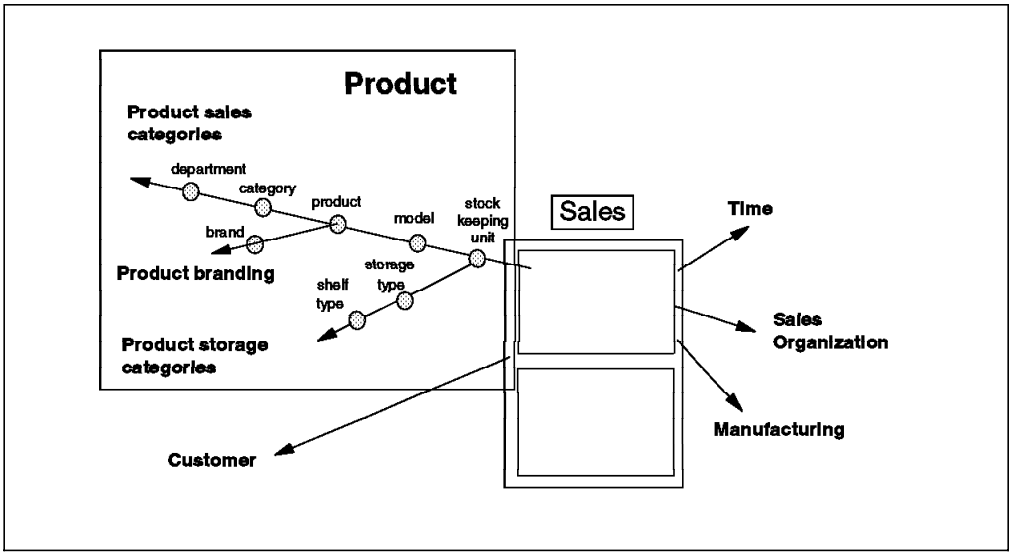


Figure 68. Sales Fact and Associated Dimensions in the Extended CelDial Case Study.

In Figure 68, some other issues related to the Product dimension are illustrated. For Sales, the model in Figure 68 provides support for aggregating sales figures not only on the stock keeping unit level (e.g., total Revenue of all sales of 4-pack Duracell Powercheck AA-batteries or IBM Thinkpad model 755CDVs) but also on the level of Packaging unit. Therefore, all Sales Revenue for Duracell AA-batteries, whether packed in 4-packs, 8-packs or 12-packs can be consolidated. Also for Sales, the Brand has now been determined to be a useful aggregation level for Sales revenue and all other Sales related measures. Using the examples above, the analyst can now calculate Sales revenue for Duracell, IBM, and all other brands being dealt with. Finally, in the Product Sales Category, the Department level has been considered useful for some analysts. This aggregation level can be used to aggregate sales revenue for such categories as Personal Computers, Video Equipment, Hifi, Accessories, and whatever else CelDial is doing business in. In addition to the extended Product Sales Category, the model in Figure 68 also incorporates another aggregation path within the Product dimension. This path is labeled ″Product storage category″ and is used to categorize sales revenue figures based on Storage Type (e.g., Packaged Items or Bulk) and Shelf Type (e.g., Stacked on Display, Sampled, Advertised Only). Such properties associated with products which are typical of the way the items are available and presented in the retail outlet can indeed have significant influence on Sales revenue and therefore be of high interest for the analyst.

Analyzing the Extended Product Dimension: The extension of the CelDial case study is only meant to be illustrative of several important issues. It should be apparent by now that when a particular group of end users and their requirements expressed as or derived from a set of business queries they have formulated, are considered in a narrow way as *the* problem to solve, the resulting solution model could well be limited to what these particular end users need. This usually leads to serious problems when new requirements surface and, in particular, when new end-user groups are identified. To make a dimensional model longer lasting and suitable for more user groups, it is of

particular importance to model the dimensions from a conceptual point of view, looking for fundamental aggregation paths and aggregation levels in which the end-user community at large may be interested. Adding measures to well defined facts usually can be done without much impact on the model at all. For the dimensions however, this is certainly not true (notice that we have not yet studied time variancy aspects within dimensions....). In the Inventory and Sales models considered before, we apparently have to deal with three dimension hierarchies within the Product dimension:

1. Product Inventory categories
2. Product Sales categories (also called the *Merchandise hierarchy*)
3. Product Storage categories

The first two dimensional hierarchies show commonalities, in fact, it looks now as though the Product Inventory categories path is a subset of the Product Sales categories path. The modeler has to decide to merge the two paths in one or keep them separate. This is largely a matter of semantics: do these paths actually represent exactly the same things or not? If so, they probably should be merged together.

You should expect to have several different aggregation paths within a single dimension, as soon as the problem domain is considered in a broad context. Splits in the dimension hierarchies can occur at different levels. Hierarchies that were split can later be split again. This process can obviously result in complex schemas, perhaps too complex for end users to deal with. Common sense should be applied, and unnecessary complications avoided.

An important and often difficult decision to make is whether an aggregation level actually is an element of a hierarchy (a structural hierarchy, that is) or whether it simply is a property of an item in the dimension. Is it, for instance, wise or required to keep the packaging unit or the brand or the storage type as explicit elements (read: potential entity types in the dimension′s hierarchies) of the dimension path or can they simply be considered properties of the products?

When investigating how exactly one has to deal with this issue in the dimension model, the modeler usually is tempted to flatten the dimension structure and ultimately reduce the whole part to a set of properties of items in the base dimension table. When doing this, the modeler is actually transforming the structured snowflake model for the dimension into a flat (or flatter) star schema.

Looking for Fundamental Aggregation Paths: Looking for fundamental aggregation paths within a dimension like the Product dimension usually means investigating a number of typical relationships within the dimension (many of these relationships are in fact relationship patterns that exist in most dimensions, by the way.):

1. Construction or structural relationships

   These relationships, usually represented as a bill of material (BOM), are used by information analysts to explore constructive relationships between products and their components. For instance, the BOM can be used to calculate the cost of a product, using the costs associated with the product′s components and the costs associated with the construction of the product.

2. Variation relationships

   Variations are used to differentiate products in terms of product models, versions, implementations, component mixtures, blends, and so on.

Variations may also be used to identify product replacements. Information analysts use variation relationships to group related products and aggregate associated measures, because the lower level categories of products may only exist for a limited period of time or because they are frequently used to replace each other in a process (e.g., when a version of a product is sold to customers when the ″original″ item is out of stock).

3. Classification relationships

   Classifications are arrangements of like products into groups. Classifications of relationships are obviously the most frequently occurring relationships between products that information analysts use to roll up detailed measures like Sales Revenue in the CelDial case. Notice that several different kinds of classifications are usually required. For example, products may be classified according to sales oriented, manufacturing-oriented, stocking-oriented, or supply-oriented characteristics. Information analysts use classifications for aggregating measures in statistical groupings such as totals, averages, minima, and maxima.

All of these relationships are obvious candidates for being modeled in the product dimension. Exactly how many of them should be incorporated in a model of the dimension is subject to the best judgment of the modeler, who will take into account the scope of the project.

The Manufacturing Dimension: CelDial′s manufacturing dimension is a typical organizational dimension of which several different variants can be found in many data warehouse models. CelDial′s Manufacturing dimension is trivially simple. It somehow incorporates the inventory organization (in other words, the physical warehouses and stocking locations), represented here by the notion of ″the plant.″ In addition, a regional aggregation facility is provided for satisfying stated end-user requirements.

In reality, the Manufacturing dimension should be investigated with the intention of establishing a solid and flexible model, representing the manufacturing organization. Two typical aspects may be important for the Manufacturing dimension, when looked upon from the viewpoint of sales analysis. If the coverage of the Manufacturing dimension would also be extended to other information analysis domains (such as analyzing costs of manufacturing activities), other important aspects would have to be added to the dimension model.

First of all, the manufacturing dimension possibly can incorporate stock keeping as it is done in the company. When analyzing the scope and content of the Manufacturing dimension, it should firmly be established to what extent the inventory and stock keeping organization should be part of this dimension or whether all of these related items should best be made into a corporate dimension.

If the company would later also show interest in setting up a detailed inventory analysis system, this data warehouse model would certainly develop an extensive Warehouse or Inventory dimension, in which parts of the manufacturing dimension could be integrated. One of the questions the CelDial data warehouse modeler is facing is determining how much generic modeling should be performed. As we indicated for the Product dimension, it usually is advantageous to make dimension models as generic as possible. This recommendation also holds for an important dimension such as CelDial′s Manufacturing dimension.

The Manufacturing dimension typically can also include cost elements associated with the manufacturing process. These cost elements may be important for the sales analysis process, because the analysts expressed their interests in Sales Revenue as well as in Profit.

The Customer Dimension: The Customer dimension, which is appropriate for a manufacturing company like CelDial, would almost certainly have two major aggregation paths or dimension hierarchies:

- One path would represent the Shipping and Billing aspects related to customers
- Another candidate path would concentrate on the Regional location of customers.

The CelDial case also includes individual customers. This usually makes the Customer dimension potentially extremely large. Such Customer dimensions need special attention. Financial service companies, retailers, and telephone companies all have such Customer dimensions, with millions of entries. Customer dimensions such as these often are used for analyzing measures and facts using demographic properties associated with customers. Examples of such properties are age, sex, marital status, number of children in the family, income levels, and education types and levels. These fields are usually heavily involved in composite selections.

In such cases, we suggest creating demographic profiles, which consist of a representative combination of demographic properties of customers. Although customers keep on having their detailed demographic properties stored in the Customer dimension, they also become associated with the demographic profiles that have been established. Using this approach can result in significant savings in terms of complexities of queries end users otherwise would have to perform as well as in substantial performance benefits. In addition, the demographic profiles establish a kind of standard classification schema applicable for customers.

The technique of creating demographic profiles can also be used for other very large dimensions in which the elements have a large number of properties that end users tend to combine in their selection criteria. The Product dimension often has such characteristics.

The Sales Organization Dimension: The Sales Organization dimension is one of the core organizational dimensions for sales-oriented information analysis. It incorporates all information required to classify and aggregate detailed sales information according to Sales persons, Sales teams, and Sales territories, districts, and regions.

The Sales Organization dimension usually is considerably smaller than the Product and Customer dimensions. It incorporates an organizational and a regional hierarchy. The Sales Organization dimension typically also includes interesting relationships with the Customer dimension, to identify territory relationships and sales responsibilities existing between Sales persons in the organization and customers. For this reason, the Sales organization dimension is sometimes incorporated as a dimension hierarchy within the Customer dimension.

The Time Dimension: Because a data warehouse is an integrated collection of historical data, the ability to analyze data from a time perspective is a fundamental requirement for every end user of the data warehouse.

Given that end users perform aggregations and roll up and drill down using various elements of time (such as day, week, month, year, fiscal year, promotion period, holiday, weekend or business day, and holiday period), we can easily see that a time dimension will have to be added explicitly to the data warehouse. Although relational databases (RDBMSs) and SQL provide some means of working with dates, times, time stamps, and durations, today's RDBMS support of time is by far not sufficient to support the kinds of aggregating and drilling activities that information analysts perform.

As a consequence, and for the very same reasons Product and Customer dimensions had to be developed and added to the dimensional model, we have to add explicitly a time dimension to the model. As we will see in this section, this time dimension provides end users with a source of structured temporal elements, relationships between these time elements, and even temporal constraints to help them in their information analysis activities. The time dimension also provides a source of row headers for reporting purposes and other textual and descriptive information about time.

In the next few sections, we develop a concise model of the time dimension.

8.4.4.2 Developing the Basis of a Time Dimension Model

Let us first consider how to set up a time dimension model that would support a granularity level of day throughout the data warehouse.

The time dimension should in the first place provide support for working with calendar elements such as day, week, month, quarter, and year. Figure 69 shows an example of a simple model that supports such elements.

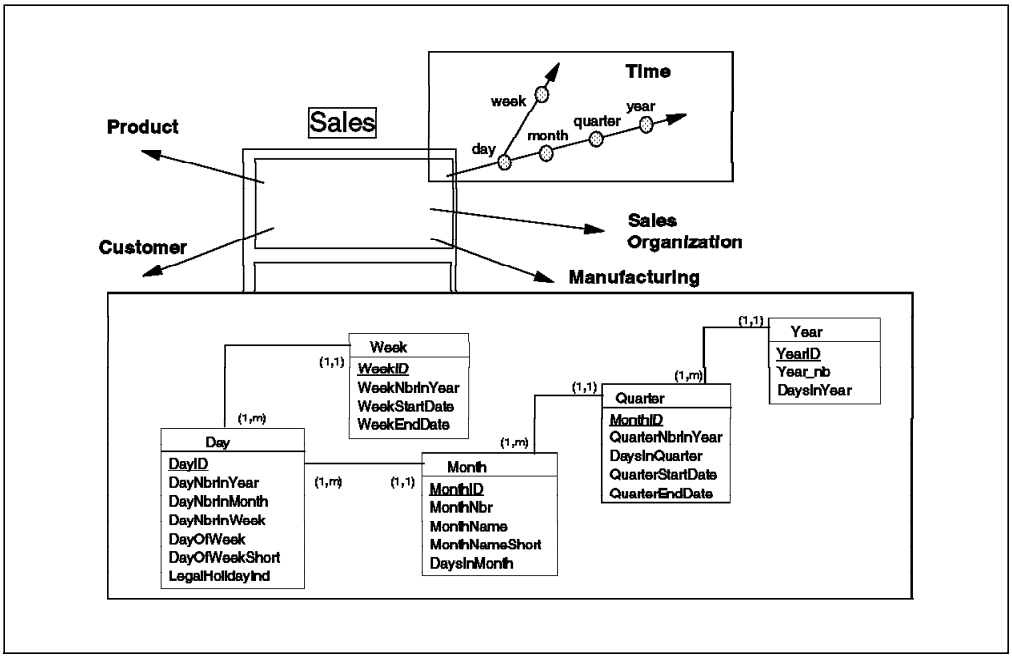


Figure 69. Base Calendar Elements of the Time Dimension.

The model in Figure 69 incorporates the following assumptions:

1. Each fact with a time granularity of day can be associated with the base time dimension element ″Day.″
2. Day is part of a Month, which is part of a Quarter, which in turn is part of a Year. This hierarchy is captured directly in the model.
3. Day is also part of a Week, but Week does not gracefully fit within any of the other calendar elements (Month, Quarter, or Year). We therefore do not have any aggregation path from Week to any of the other (higher level) time dimension elements.
4. Each of the time dimension elements in the model has a unique key, which should preferably be an absolute number that can be used to identify the elements at occurrence level. This can be achieved by setting a fixed reference date as the ″starting point″ for numbering all other temporal elements.
5. All of the elements in the time domain are provided with (many) calendar-related properties.

Users can now query any fact table that can be associated with the time dimension, asking for total Sales Revenue on a particular day, say 7/31/97; roll up per Month, for instance, Total Sales Revenue for the month of July 1997; and further roll up to total Revenue for the year 1997. Similarly, rolling up daily measures to week totals or averages is also possible.

Using the attributes of the calendar elements in this time dimension model, end users can further explore temporal aspects of facts, such as selecting all Sales facts that can be associated with Tuesdays of the first Quarter of 1996.

About Aggregation Paths above Week: Considering the Week element in the time dimension and, in particular, possible aggregation paths from Week to either Month, Quarter, or Year leads us to a peculiar problem with parallel aggregation paths in general. The base issue is illustrated in Figure 70 on page 129, where possible aggregation paths are considered between Week and Year (from Week to Month or from Week to Quarter leads to similar considerations).

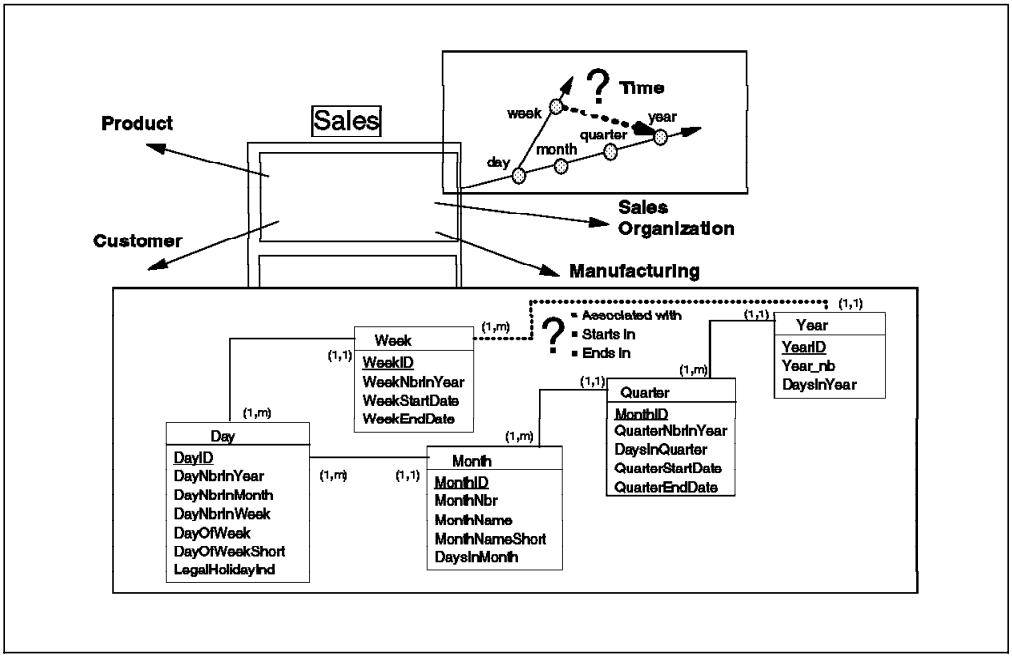


Figure 70. About Aggregation Paths from Week to Year.

The issue stems from the fact that weeks do not fully fit within years, as opposed to days. A year consists of 52+ weeks, and, as a matter of fact, every six years the year consists of 53 weeks. The time dimension modeler therefore has to decide which relationship between Week and Year can be and should be incorporated in the model. One possible solution is to associate each week with the year in which it starts. A similar solution would be to associate each week with the year in which it ends. Other solutions may also be feasible.

If any of these associations between Week and Year is present in the model, end users will have to be made aware of it because they obviously have to know the precise semantics of the association.

There is more that end users will have to know, however. If any of these associations is present in the time dimension model, end users must be made aware of the fact that an aggregation performed with the Day-Month-Quarter-Year path will yield different results from those of an aggregation performed on the same base facts with the Day-Week-Year path. To verify this statement, try to aggregate Sales Revenue using the Day-Month-Quarter-Year path and the Day-Week-Year path. In principle, the values you get will differ.

There are a couple of ways to deal with this situation. One of the solutions consists of simply informing the end users and telling them not to switch paths when aggregating from the Day level. Another solution consists of deleting the Week-Year association. Now end users can no longer really aggregate from Week to Year. A third solution consists of adding a new temporal element in the time dimension, which could be called the *Weekly Reporting Year*.

As a ground rule, if two parallel consolidation paths within a dimension have a common start and end point, the modeler should be absolutely sure that aggregating measures along any such paths would result in exactly the same aggregated results. If not, either of the three alternative solutions illustrated above should be considered for the model. Although we brought up the

modeling issue when dealing with the time dimension, the problem of cyclic paths in a dimensional model is a general problem. It may occur in other dimensions as well.

Business Time Periods and Business-Related Time Attributes: In the time dimension model of Figure 69 on page 127, only temporal elements and time attributes are included, with a fixed definition and meaning within a given calendar. Organizations have their own business time periods, however, such as fiscal years, seasons, promotion periods, and holiday periods. In addition, several time-related attributes have a precise meaning only within a given organizational context. Figure 71 illustrates candidate business time periods and business-related time attributes that should further be added to the time dimension model.

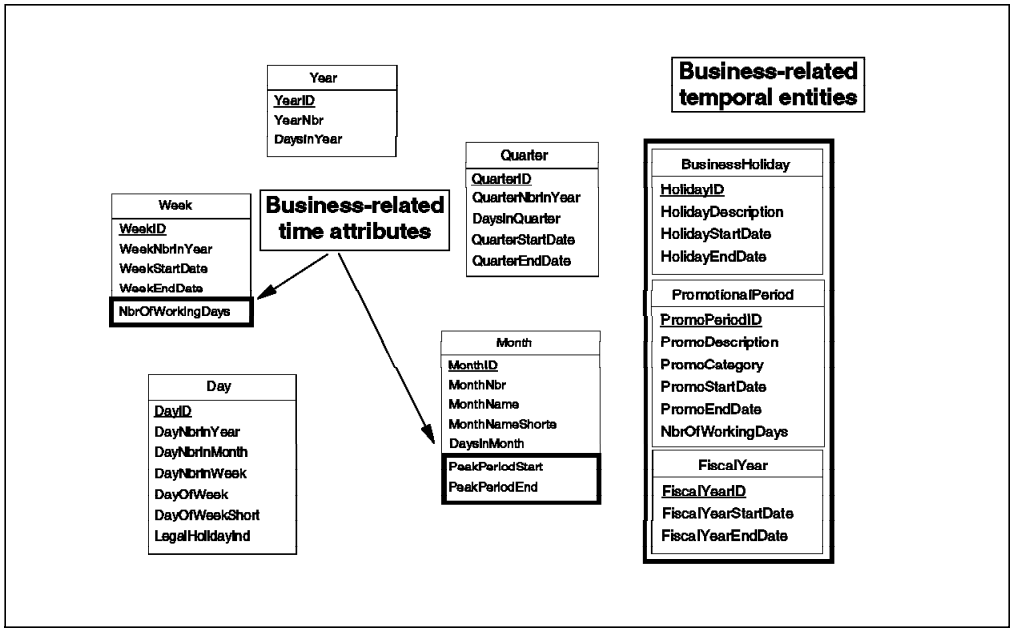


Figure 71. Business-Related Time Dimension Model Artifacts.

The result of adding these elements and attributes to the time dimension model is illustrated in Figure 72 on page 131.

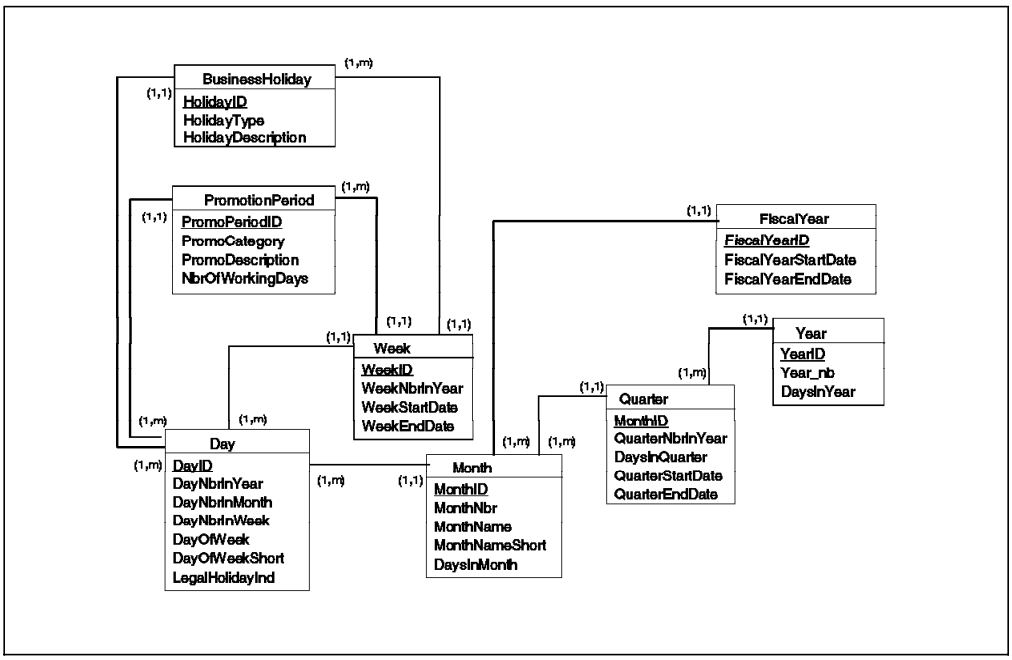


Figure 72. The Time Dimension Model Incorporating Several Business-Related Model Artifacts.

Making the Time Dimension Model More Generic: The time dimension model in Figure 72 is becoming quite complex. In addition, the model itself is not very resilient to changes in the area of business-related time periods. Knowing that new kinds of time periods will constantly be defined, we can say that the model in Figure 73 results in a better solution. With the supertype/subtype modeling technique, promotion periods, holiday periods, and other business-related time periods with a substantial amount of similar attributes and relationships with other elements of the time dimension are modeled as subtypes of a more generic element of the model called *Business Period*. If, in addition, the business period types are added to the model, the structure becomes much more flexible and simple.

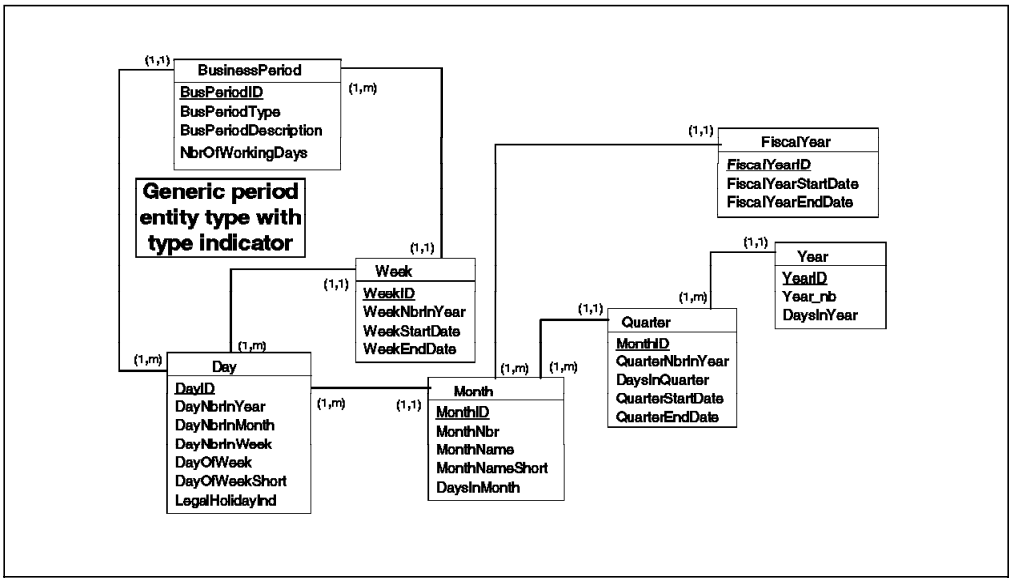


Figure 73. The Time Dimension Model with Generic Business Periods.

Flattening the Time Dimension Model into a Dimension Table: Figure 74 on page 132 illustrates the effects of transforming the structured time dimension model of Figure 73 into a flat dimension table. The result is quite tricky, when done for this kind of model.

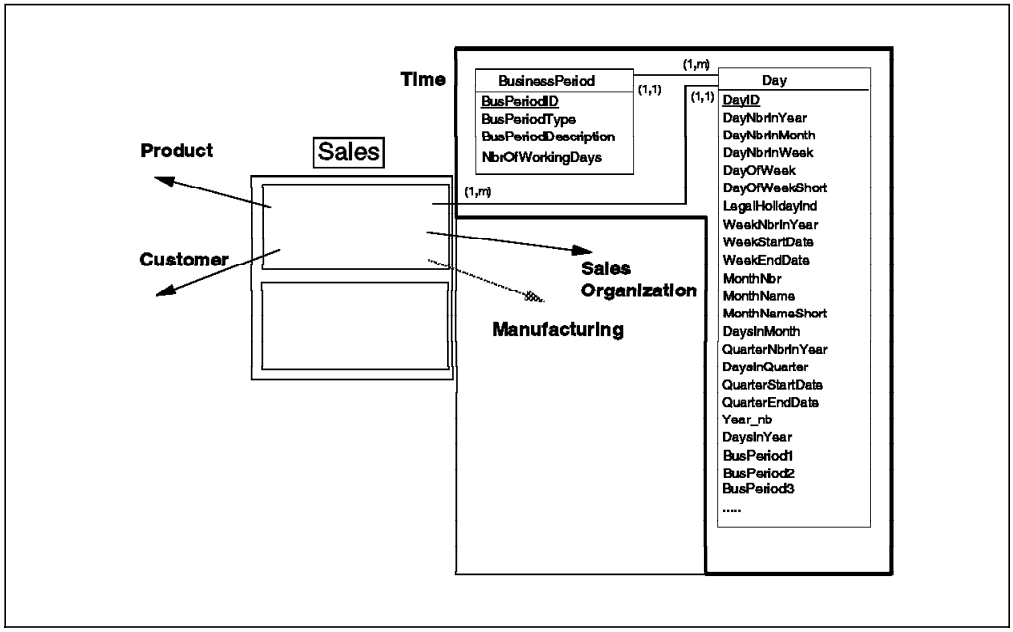


Figure 74. The Flattened Time Dimension Model.

When comparing the structured and flat time dimension models in Figure 73 on page 131 and Figure 74, it becomes apparent that end users will have to be much more considerate when they roll up through the flat time dimension. The flat structure, although easy to understand and query, requires a good understanding of the implicitly incorporated hierarchies within it. The snowflake structure documents the explicit structure of the hierarchies and is thus less likely to be misinterpreted. However, the structured snowflake schema definitely is more complex to understand, and it will obviously also be more time consuming to explore the relationships between the modeling elements.

The Time Dimension As a Means for Consistency: The company as a whole must agree on precise definitions and specifications of the time-related elements used by the various information analysts. The time dimension can be of great help to achieve this kind of companywide consistency. Ideally, within the whole data warehouse, there is only one time dimension, which reflects exactly how the company deals with its time-related aspects of information analysis.

The time dimension is densely packed with attributes and relationships. This density becomes even more pronounced if several countries or cultures are involved and there are wide variations of business periods and business-related time attributes within the company. In such cases, the corporatewide time dimension may be developed and populated centrally and be made available for local or departmental use. Departmental or even lower-level time dimensions should then be derived from the corporate time dimension, and these should include exactly those modeling elements, relationships, and attributes that the end users require.

This way or organizing the time dimension in two modeling ″layers″ (one complex corporate time dimension with several local time dimensions derived

from it) usually is an ideal solution. The simpler local time dimensions also are more suitable for being flattened into a time dimension table. In this way, the performance and querying capabilities of the total solution are further maximized.

Notice that in the absence of a corporatewide time dimension, every end-user group or every department will develop its own version of the time dimension, resulting in unlike meanings and different interpretations. Because time-related analysis is done so frequently in data warehouse environments, such situations obviously provide less consistency.

Lower Levels of Time Granularity: Depending on specific business organization aspects and end-user requirements, the granularity of the time dimension may have to be even lower than the day granularity that we assumed in the previously developed examples. This is typically the case when the business is organized on the basis of shifts or when a requirement exists for hourly information analysis.

8.4.4.3 Modeling Slow-Varying Dimensions

We have investigated the time dimension as a specific dimension in the data warehouse and have assumed that dimensions are independent of time. What we now need to investigate is how to model the temporal aspects in the dimensions of the dimensional data model. Dimensions typically change slowly over time, in contrast to facts, which can be assumed to take on new values each time a new fact is recorded. The temporal modeling issues for dimensions are therefore different from those for facts in the dimensional model and consequently also the modeling techniques, commonly referred to as *modeling techniques for slow-varying dimensions*.

When considering slow-varying dimensions, we have to investigate aspects related to keys, attributes, hierarchies, and structural relationships within the dimension. Key changes over time are obviously a nasty problem. Changes to attributes of dimensions are less uncommon, but special care has to be taken to organize the model well so that attribute changes can be recorded in the model without causing (too much) redundancy. Structural changes also occur frequently and must be dealt with carefully. For example, a product can change from category X to category Y, or a customer can change from one demographic category into another.

About Keys in Dimensions of a Data Warehouse: Keys in a data warehouse should never change. This is an obvious, basic tenet. If it is not met, the data warehouse′s ability to support analysis of yesterday′s and today′s data, say, 10 years from now, producing the same results as we get today, will be hampered. Likewise, if keys in the data warehouse change, it will soon become difficult to analyze the data in the data warehouse over long periods of time.

Making keys in a data warehouse time-invariant is a nasty problem, however, involving a number of specific issues and considerations related to the choice of keys and to their generation and maintainability. Figure 75 on page 134 depicts one example of the effects of time variancy on keys. In that example, we must capture the event history, but it needs to reflect state history. In this case, we add fields to reflect the date and duration of state changes.

Data moved into a data warehouse typically comes from operational application systems, where little or no history is kept. OLTP applications perform insert,

update, and delete operations against database records, thereby creating key values and destroying them. Even updates of key values may occur, in which case the new key values may represent the same objects as before the change, or they may represent new ones. When data records are inserted in the OLTP database and consequently when key values for the inserted records are established, these values may be new ones or reused ones. If key values are being reused, we will have to find a solution for these keys in the data warehouse environment, to make sure the history before the reuse took place and the history after the reuse are not mistakenly considered to be part of a single object's lifespan history.

Yet another typical issue with keys in a data warehouse is when data for a particular object comes from several different source data systems. Each system may have its own set of keys, potentially of totally different format. And even if they would have the same format, a given key value in one source system may identify object ABC while in another system it could identify object XYZ.

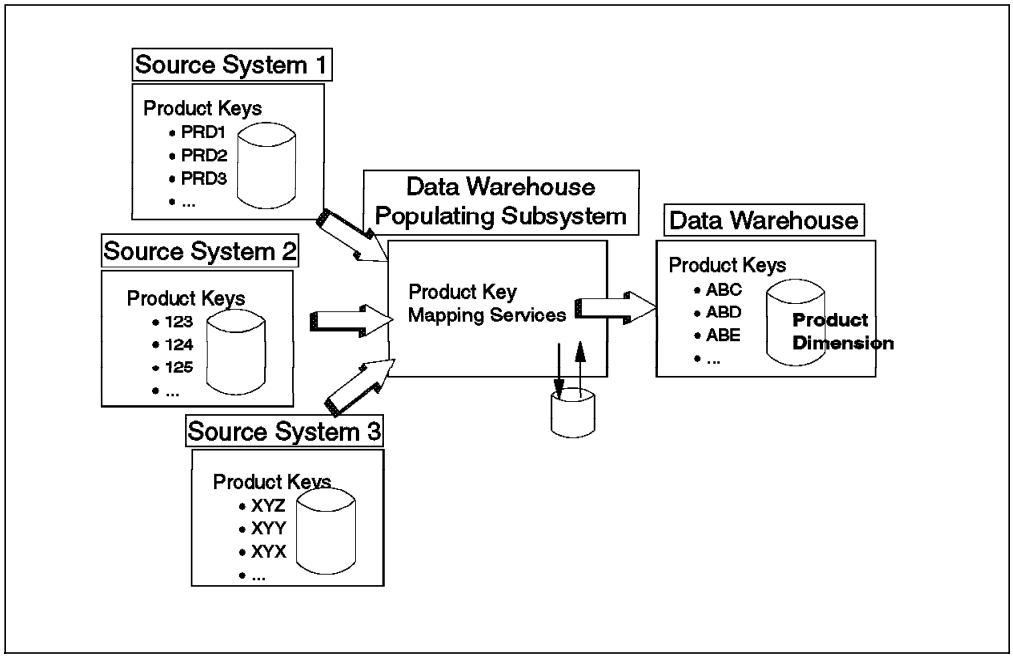


Figure 75. Time Variancy Issues of Keys in Dimensions.

Based on these observations, we can no longer expect to be able to take the simple solution and keep the OLTP source data keys as the data warehouse keys for related objects. The simple trick may work, but in many cases we will have to analyze and interpret the lifespan history of creation, update, and deletion of records in the source database systems. Based on this analysis of the lifespan history of database objects, we will have to design clever mechanisms for identifying data warehouse records and their history recordings.

Typical elements of a key specification mechanism for a data warehouse are:

- A mechanism to identify long-lasting and corporatewide valid identifiers for objects whose history the data warehouse will record. These identifiers may be external or internal (system-generated) identifiers. Internal identifiers are obviously difficult to use by external end users. They can be used only internally to associate the different records and subrecords that make up the object's representation in the data warehouse. One possible technique

consists of concatenating the object's key in the OLTP source database (if suitable for the data warehouse environment) with the record's creation time stamp. More complex solutions may be required.

- Techniques to capture or extract the source database records and their keys and translate them mechanically into the chosen data warehouse keys. The technique mentioned above, consisting of concatenating the OLTP key with the creation time stamp, is rather easily achievable if source data changes are captured. We may have to deal with more complex situations; in particular, having to provide key value translations, using lookup tables, is a common situation. Notice too that if lifespan histories are important for transforming key values for the data warehouse, it must be possible to capture and interpret the lifespan activities that occur in the OLTP source systems. It obviously makes no sense to design a clever key mechanism based on recognizing inserts, updates, and deletes, if these operations cannot consistently and continuously be captured.
- The mechanism of key transformations will have to be extended with key integration facilities, if the records in the data warehouse are coming from different source application systems. This obviously increases the burden on the data warehouse populating subsystem.
- When keys are identified and the key transformation system is established, it is good practice to do a stability check. The designer of the key system for the data warehouse should envisage what happens with the design specifications if operational systems are maintained, possibly involving changes to the source system's key mechanism or even to its lifespan history. Another important aspect of this stability check would be to investigate what happens if new source application systems have to be incorporated into the data warehouse environment.

The issues about keys discussed above are typical for data warehouses. They should be considered very carefully and thoughtfully as part of the activities of modeling the slow-varying dimensions. The solutions should be applicable too for keys in the fact tables within the dimensional model. Keys in fact tables are most frequently foreign keys or references to the primary identifier of data warehouse objects, as they are recorded in the dimensions. Notice too that dimension keys should preferably not be composite keys, because these cause difficulties in handling the facts.

Because data marts usually hold less long-lasting history (frequently, data marts are temporal snapshots), the problems associated with designing keys for a data mart may be less severe. Nevertheless, the same kinds of considerations apply for data marts, especially if they are designed for a broad scope of usage.

In 8.4.4.4, "Temporal Data Modeling" on page 139 we develop more techniques for transforming nontemporal data models (like the dimensional models we have developed so far) into temporal models suitable for representing long-lasting histories. For those of you not fully familiar with the issues mentioned here, that section will help you further understand the types of problems and the techniques for handling them.

Dealing with Attribute Changes in Slow-Varying Dimensions: The kind of problems we have to deal with here can be illustrated as follows. Operational applications perform insert, update, and delete operations on the source databases and thereby replace the values that were previously recorded for a

particular object in the database. Operational applications that work in that way do not keep records of the changes at all. They are inherently nontemporal.

If such source databases are extracted, that is, if a snapshot of the situation of the database is produced, and if that snapshot would be used to load the data warehouse's dimensions, we would have inherently nontemporal dimensions. If a product in the product dimension would be known to have the color red before the snapshot of the product master database is loaded in the data warehouse, that product could have any color (including its previous color red) after the snapshot is loaded. Slow-varying dimension modeling is concerned with finding solutions for storing these attribute changes in the data warehouse and making them available to end users in an easy way (see Figure 77 on page 138).

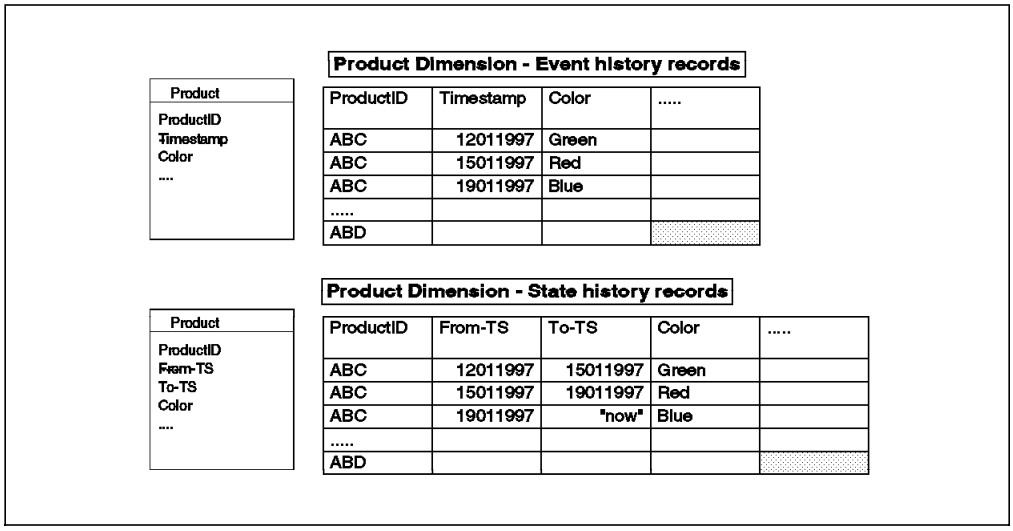


Figure 76. Dealing with Attribute Changes in Slow-Varying Dimensions.

What we have to do, using the previous example of a product and its attribute *color* in the product dimension, is record not only the value for the product's color but also when that value changes or, as an alternative solution, record during which period of time a particular value for the color attribute is valid. To put it differently, we either have to capture the changes to the attributes of an object and record the full history of these changes, or we have to record the period of time during which a particular value for an attribute is valid and compile these records within a continuous recording of the history of the attribute of an object.

With the first approach, called *event modeling*, data warehouse modeling would enable the continuous recording of the changes that occurred to the product's color, plus the time when the change took place.

The second approach, called *state modeling*, would produce a model for the slow-varying product dimension which would enable the recording of the product's color plus the period of time during which the particular color would be valid.

Both event and state modeling approaches are viable techniques for modeling slow-varying dimensions. As a matter of fact, both techniques can be mixed within a given dimension or across the dimensions of the data warehouse. Deciding which technique to use can be done considering the prime use of data in the dimension: If there is more frequent interest in knowing when a particular

value was assigned to an attribute, an event modeling technique is naturally fitting. If there is more frequent interest in knowing when or how long a particular value of an attribute is valid, a state modeling approach is probably more suitable. For data marts with which end users are directly involved, this decision will be somewhat easier to make than in cases where we do dimension modeling for corporate data warehouses.

Notice that change events can be deduced from state models only by looking at when a particular value for an attribute became valid. In other words, to know when the color changed, if the product dimension is modeled with a state modeling technique for the color attribute, just look at the begin dates of the state recordings. Likewise, the validity period of the value of an attribute, for example, the color red, can be deduced from an event model. In this case, the next change of the attribute must be selected from the database, and the time of this event must be used as the end time of the validity period for the given value. For example, if you want to find out during which period the color of a given product was red, look for the time the color effectively turned red first and then look for the subsequent event that changed the color. It is clear that querying and performance characteristics of the two cases are not at all the same. That is why the choice of modeling technique is driven primarily by information analysis characteristics.

Modeling of slow-varying dimensions usually becomes impractical if the techniques are considered on an attribute level. What is therefore required are techniques that can be applied on records or sets of attributes within a given database record. In 8.4.4.4, “Temporal Data Modeling” on page 139, we show exactly how this can be performed.

Modeling Time-Variancy of the Dimension Hierarchy: We have not discussed at all how to handle changes in the dimension′s hierarchy or its structure. So let′s investigate what happens to the model of the dimension if changes occur that impact the dimension hierarchy (see Figure 78 on page 139).

At first, there seem to be two issues that need to be looked at. One is where the number of levels in the hierarchy stay the same, and thus only the actual instance values themselves change. The other is when the number of dimension hierarchy levels actually changes, so that an additional hierarchy level is added or a hierarchy level is removed.

Let′s consider first the situation when a hierarchy instance value changes. As an example, consider the situation where the Category of Product ABC changes from X into Y. Notice we also want to know when the change occurred or, alternatively, during which period Product ABC belonged to Categories X or Y.

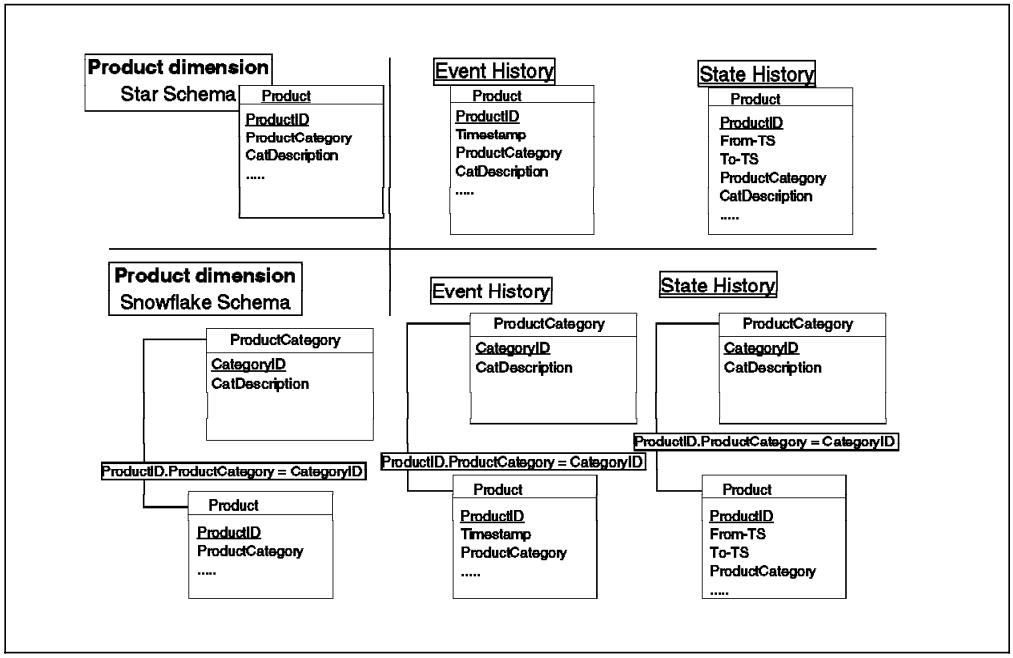


Figure 77. Modeling Time-Variancy of the Dimension Hierarchy.

In a star schema, the category of Product ABC would simply be one of the attributes of the Product record. In this case, we obviously are in a situation that is identical to the attribute situation, described in the previous section. The same solution techniques are therefore applicable.

If a snowflake modeling approach for the Product dimension would have been used, the possible product categories would have been recorded as separate records in the dimension, and the category of a particular product would actually be determined by a pointer or foreign key from the product entry into the suitable Category record. To be able to capture the history of category changes for products in this case, the solution would consist of capturing the history of changes to the foreign keys, which again can be done using the same attribute-level history modeling techniques described above.

A somewhat bigger issue for modeling slow-varying dimensions is when there is a need to consider the addition or deletion of hierarchy levels within the dimension. The solution depends on whether a star or a snowflake schema is available for the dimension. In general though, both situations boil down to using standard temporal modeling techniques.

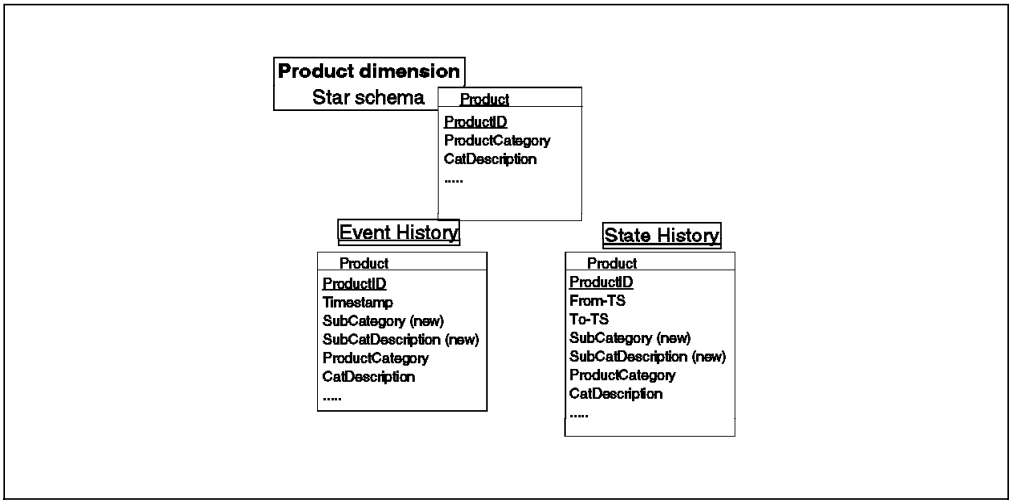


Figure 78. Modeling Hierarchy Changes in Slow-Varying Dimensions.

For dimensions with a flat star model, adding or deleting a level in a hierarchy is equivalent to adding or deleting attributes in the flat dimension table that represent the hierarchy level in that dimension. To solve the problem, the modeler will have to foresee the ability to either add one or more attributes or columns in the dimension table or to drop the attributes. In addition to these changes in the table structure, the model must also make room for adding time stamps that express when the columns were added or dropped.

For dimensions with a snowflake schema, adding a dimension level or deleting one must be modeled as a change in the relationships between the various levels of the hierarchy. This is a standard technique of temporal data modeling.

As soon as the data warehouse begins to support requirements related to capturing structural changes in the dimension hierarchies, including keeping a history of the changes, end users will be facing a considerably more complex model. In these cases, end users will need more training to understand exactly how to work with such complex temporal models, analyze the data warehouse, and exploit the rich historical information base that is now available for roll up and drill down. How exactly to deal with this situation depends to a large extent on the capabilities of the data analysis tools.

8.4.4.4 Temporal Data Modeling

Temporal data modeling consists of a collection of modeling techniques that are used to construct a temporal or historical data model. A temporal data model can loosely be defined as a data model that represents not only data items and their inherent structure but also changes to the model and its content over time including, importantly, when these changes occurred or when they were valid (see Figure 79 on page 140). As such, temporal or historical data models distinguish themselves from traditional data models in that they incorporate one additional dimension in the model: the time dimension.

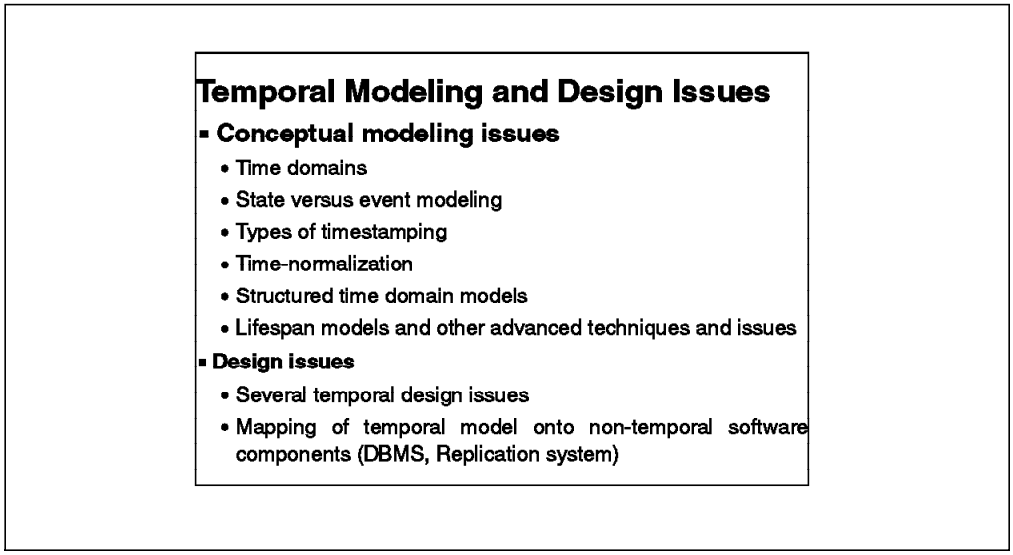


Figure 79. Adding Time As a Dimension to a Nontemporal Data Model.

Temporal data modeling techniques are required in at least two important phases of the data warehouse modeling process. As we have illustrated before, one area where these techniques have to be applied is when dealing with temporal aspects of slow-varying dimensions in a dimensional model. The other area of applicability for temporal data modeling is when the historical model for the corporate data warehouse is constructed. In this section, we exploit the basic temporal modeling techniques from a general point of view, disregarding where the techniques are used in the process of data warehouse modeling. Notice that temporal modeling requires a lot more careful attention than just adding time stamps to tuples or making whole sections of data in the data warehouse dependent on some time criterion (as is the case when snapshots are provided to end users). Temporal modeling can add substantial complexity to the modeling process and to the resulting data model.

In the remainder of this section, we use a small academic sample database called the Movie Database (MovieDB) to illustrate the techniques we cover. Notice that the model does not include any temporal aspects at all, except for the ″Year of Release″ attribute of the Movie entity (see Figure 81 on page 142).

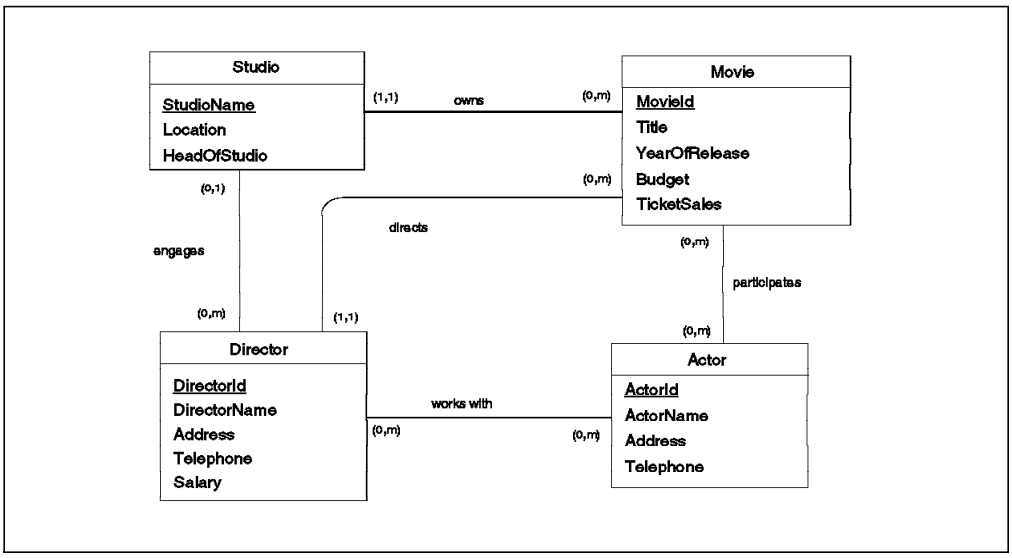


Figure 80. Nontemporal Model for MovieDB.

Let us assume that an ER model is available that represents the model of the problem domain for which we would like to construct a temporal or historical model. This is for instance the situation one has to deal with when modeling the temporal aspects of slow-varying dimensions: the dimension model is either a structured ER model, when the dimension is part of a snowflake dimensional model, or a flat tabular structure (in other words, coincides with a single entity) when the dimension is modeled with a star modeling approach.

Likewise, when the corporate data warehouse model is constructed, either a new, corporatewide ER model is produced or existing source data models are reengineered and integrated in a global ER schema, which then represents the information subject areas of interest for the corporate data warehouse. Temporal data modeling can therefore be studied and applied as a model transformation technique, and we develop it from that perspective in the remainder of this section.

Preliminary Considerations: Before presenting temporal modeling techniques, we first have to review some preliminary considerations. As an example, a number of standard temporal modeling styles or approaches could be used. Two of the most widely used modeling styles are *cumulative snapshots* and *continuous history models* (see Figure 81 on page 142 and Figure 82 on page 143).

A database snapshot is a consistent view of the database, at a given point in time. For instance, the content of a database at the end of each day, week, or month represents a snapshot of the database at the end of each day, week, or month.

Temporal modeling using a cumulative snapshot modeling style consists of collecting snapshots of a database or parts of it and accumulating the snapshots in a single database, which then presents one form of historical dimension of the data in the database. If the snapshots are taken at the end of each day, the cumulative snapshot database will present a perception of history of the data in this database, consisting of consecutive daily values for the database records. Likewise, if the snapshots are taken at the end of each month, the historical perspective of the cumulative snapshots is that of monthly extracted information.

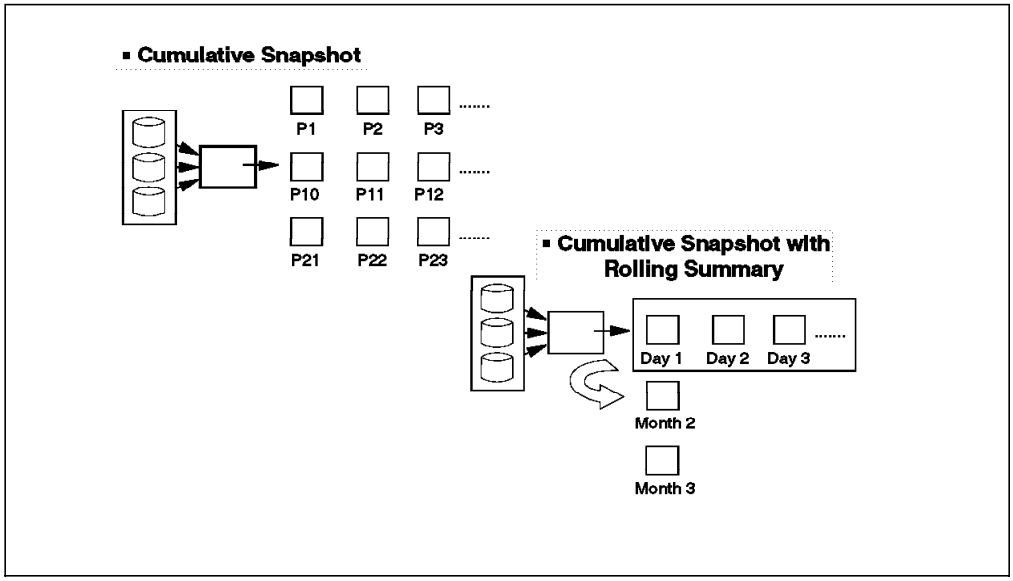


Figure 81. Temporal Modeling Styles.

The technique of cumulative snapshots is often applied without considering a temporal modeling approach. It is a simple approach, for both end users and data modelers, but unfortunately, it has some serious drawbacks.

One of the drawbacks is data redundancy. Cumulative snapshots do tend to produce an overload of data in the resulting database. This can be particularly nasty for very large databases such as data warehouses. Several variants of the technique are therefore common practice in the industry: snapshot accumulation with rolling summaries and snapshot versioning are two examples.

The other major drawback of cumulative snapshot modeling is the problem of information loss, which is inherent to the technique. Except when snapshotting transaction tables or tables that capture record changes in the database, snapshots will always miss part of the change activities that take place within the database. No variants of the technique can solve this problem. Sometimes, the information loss problem can be reduced by taking snapshots more frequently (which then tends to further increase the redundancy or data volume problem), but in essence, the problem is still there. The problem can be a serious inhibitor for data warehousing projects. One of the areas where snapshotting cannot really produce reliable solutions is when full lifespan histories of particular database objects have to be captured (remember the section, “About Keys in Dimensions of a Data Warehouse” on page 133, covering issues related to keys in dimensions of a data warehouse).

The continuous history model approach aims at producing a data model that can represent the full history of changes applied to data in the database. Continuous history modeling is more complex than snapshotting, and it also tends to produce models that are more complex to interpret. But in terms of history capturing, this approach leads to much more reliable solutions that do not suffer from the information loss problem associated with cumulative snapshots.

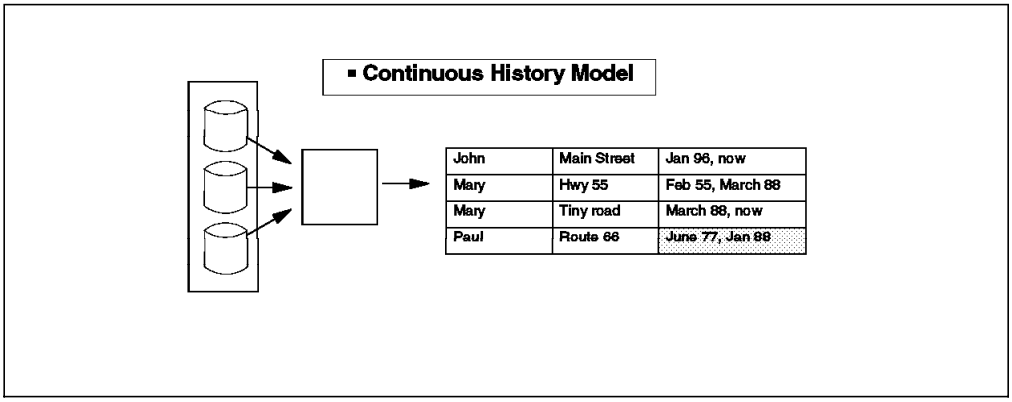


Figure 82. Continuous History Model.

In the remainder of this section, we explore techniques for temporal modeling using a continuous history modeling approach.

Time Stamp Interpretations: One of the first things you will do as a data modeler involved in temporal modeling is consider the addition of time stamps to the records of the database. In most cases, the first thing to do will actually be to look for existing time stamps or time-related attributes in records, such as dates. As a modeler, however, you should be aware of the fact that time stamps or date attributes in records may have different interpretations. Interpreting time stamps correctly and making sure the right time stamps are available in records of the historical data model are important activities.

Time stamps can indeed have several different meanings (see Figure 82). Some time stamps can represent what is known as ″valid times,″ in which case they are representative of what happens in real life. Some examples of valid time time stamps are: dates recorded on documents (the assumption is that the dates are recorded correctly), time stamps created through point-of-sales recording or scanning or through process controllers directly attached to robots or other manufacturing equipment.

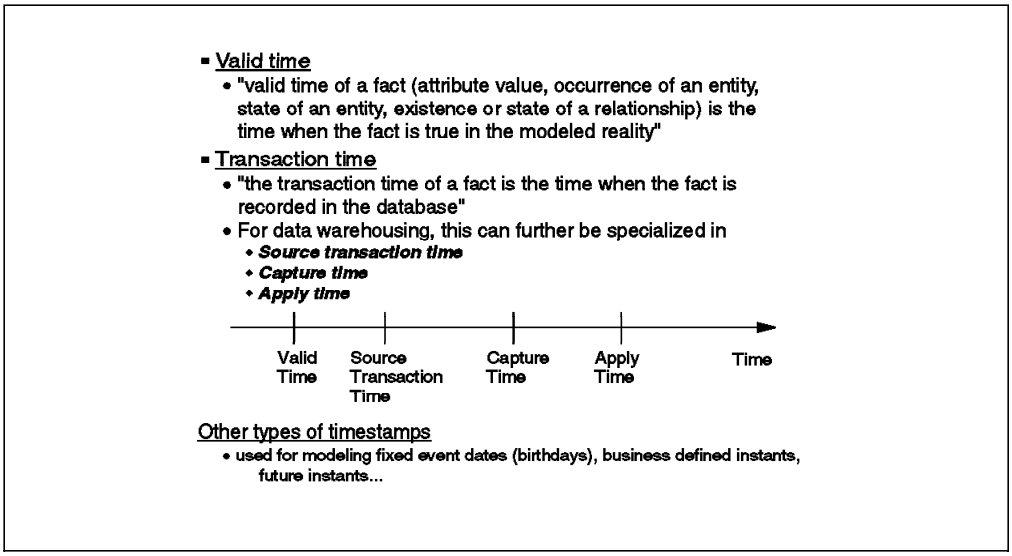


Figure 83. Different Interpretations of Time.

Other frequently encountered time stamp interpretations are:

- Transaction time: representing the time at which the operational system performed the transaction associated with a real life activity
- Capture time: the time stamp that represents the time at which the record was extracted or captured from the source database
- Apply time: the time stamp associated with the time at which the record was loaded in the data warehouse.

It is important to stress that these time stamps can be fundamentally different. Since analysis of data in a data warehouse frequently involves selecting, comparing, and/or joining records together on the basis of time stamps, care should be taken when time stamps with different interpretations (or date columns of any sort, to give just a simpler example) are used in these analysis operations. Time stamps in records belong to a particular domain, just as any other attribute in a model or any column in a table of a relational database. And we all know (do we?) that elements drawn from different domains should in principle not be used in joins and comparisons, unless there is some mapping mechanism provided. Unfortunately, database systems usually do not support these kinds of considerations (unless user-defined data types and object-relational extensions would be applied), and data modelers tend to overlook this problem.

We suggest that you pay particular attention to choosing the right interpretations of time stamps in the historical data model and that these interpretations be made available to end users through the metadata. Records can contain several different time stamps, drawn from different domains.

Instant and Interval Time Stamps: Two basic techniques are available for adding time stamps to records: instant time stamps and interval time stamps (see Figure 84).

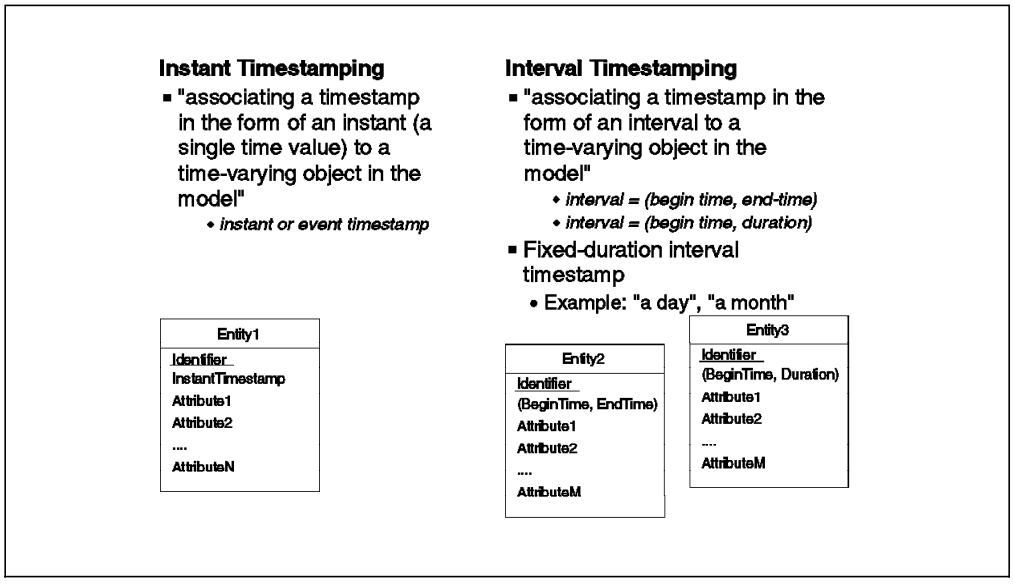


Figure 84. Instant and Interval Time Stamps.

Instant time stamps are single-valued time stamps that represent an instance of time. When an instant time stamp is added to a record, the record basically represents a transaction or an event that occurred or a data change being captured from the operational application system.

Interval time stamps represent a period of time or a time interval. Interval time stamps are in principle two-valued. They consist of either a begin time and an end time or a begin time and a duration. Interval time stamps are used basically for modeling the status of a record during a given period. A little complication comes up when intervals have a fixed duration, such as a day or a month. In this case, an apparently single-valued time stamp (a date or an attribute that represents a month for instance) may actually represent a period rather than an instant of time.

The difference between the two is not trivial. Consider for instance the following two situations, where the data models in fact are basically identical. Situation 1 provides a model for ticket sales associated with movies where each sales transaction is recorded, incorporating the number of tickets sold and the day the sale took place. In this case, the numbers in the ticket sales column of the database may freely be added across time, for instance, to calculate the total amount of tickets sold for the whole day or per month.

Situation 2 takes another temporal modeling approach. Here, the ticket sales are accumulated in a source application, and once per day (or once per hour if you wish), the value of ticket sales is extracted from the database and recorded in the temporal database, together with the day the extract took place. This is apparently the same temporal model as the previous solution, but now, it does not make sense to add the ticket sales per day or per month.

The basic difference between the two models is in the interpretation of the time stamp in the two records: in the first solution, the time stamp is an instant timestamp and the record represents actual ticket sales. In the second solution, the time stamp is in fact a (fixed duration) interval time stamp. Ticket sales in the second solution are inherently nonadditive over time.

Base Temporal Modeling Techniques

Adding Time Stamps to Entities: To kick off temporal modeling, time stamps are added to the entities of the base model. This process involves the following activities (illustrated with the MovieDB sample in Figure 85):

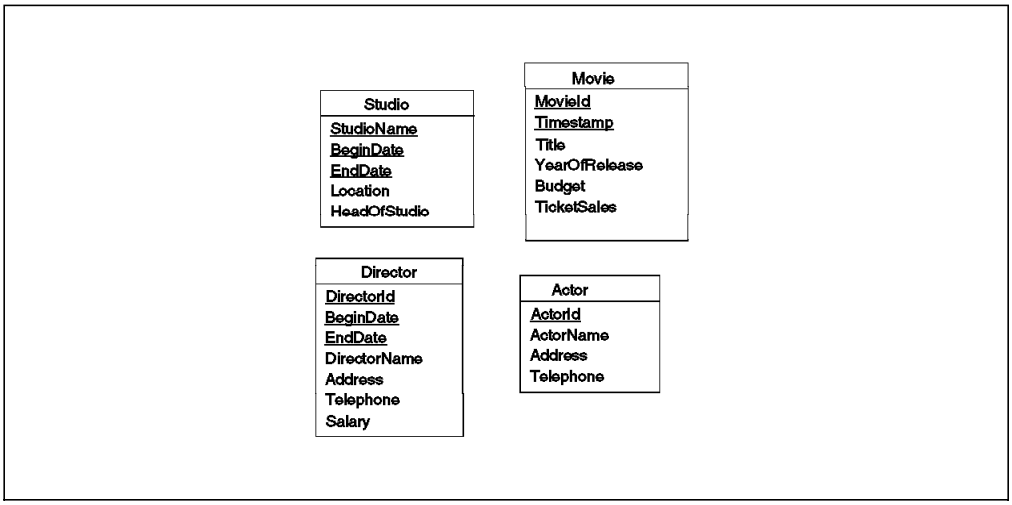


Figure 85. Adding Time Stamps to the MovieDB Entities.

1. Determine whether entities contain ″useful″ time stamps. If so, make sure you capture the precise interpretation of these attributes.

2. Add time stamps to entities, either because they do not have a usable time stamp or because they do not have a time stamp with the suitable interpretation. In the process of adding time stamps, you have to determine whether the entity will be modeled using an event or state record.

   The difference between an event and a state record is of fundamental importance. Event records are used to model transactions, events that happen in the real world, data changes that are captured, etc. State records are used to represent the status of attributes during a given period, the period being represented by the interval time stamp which is in the record.

   Event records should be given instant time stamps (inherently single-valued). State records should be given an interval time stamp (two-valued, or, if it involves a fixed-duration period, single-valued).

   In the example above, two entities are modeled using state records (Studio and Director), the Movie entity is modeled using an event record.

3. You can now also decide for which entities no history is to be kept at all. For those entities, time stamps must not be added. In the MovieDB sample, we have apparently decided not to keep track of the history of Actor.

4. Use a particular time stamp format across all of the entities that belong to the same temporal domain. This may have the effect that some records are given more than one time stamp.

Notice that relationships are momentarily not considered in this process.

Restructuring the Entities: A technique based on the principles of time-normalization can now be used to restructure the entities. The technique consists of the following activities (applied on each of the entities):

Volatility Classes. Divide the attributes in so-called volatility classes. A volatility class is a collection of attributes of an entity that are considered to change together over time. Notice that, in practice, determining volatility classes will probably require you to accept some compromises as to the likelihood that attributes do indeed change together. Volatility classes should not be too small (that is, volatility classes with only one or a few attributes are not very practical). However, if attributes within a single volatility class have unrelated change patterns, the resulting data model will include redundant information (see Figure 86 on page 147).

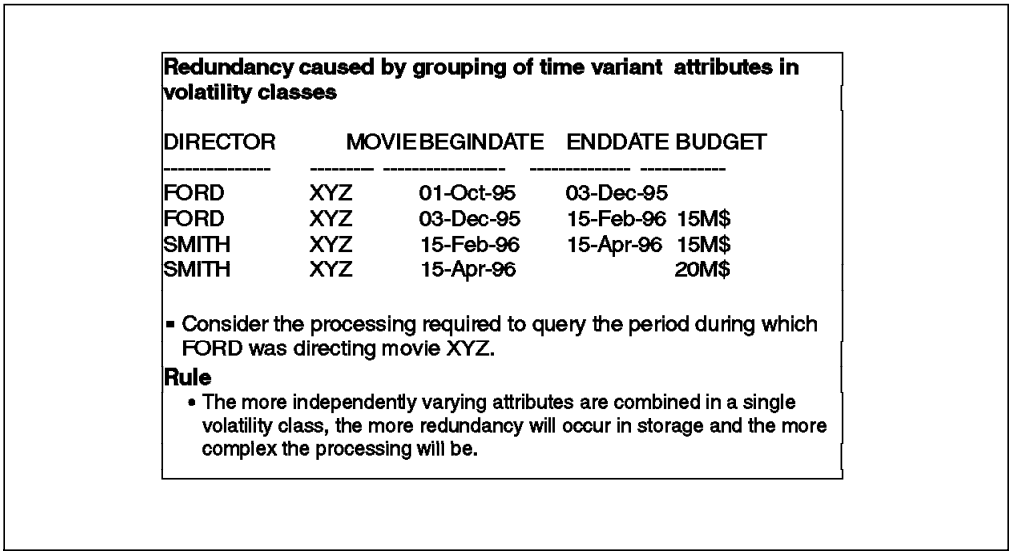


Figure 86. Redundancy Caused by Merging Volatility Classes.

Time-Invariant Volatility Class. One of the volatility classes can represent time-invariant attributes. Because keys are supposed to be time invariant, they should be part of this class of attributes. All other attributes that are invariant or are considered to be invariant (those whose history we decide not to keep track of) should be made part of this class.

Volatility classes will be mapped onto records that represent parts of the original entities. The time-invariant class will provide a very useful ″anchor″ record, which can be used to relate all other history records of the original entity.

The time-invariant volatility class for the entity consists of DirectorID (the key), DirectorName, Address, and Telephone (we have determined we are not interested in the history for these). Notice that history records can be modeled as dependent entity types.

Time-Variant Volatility Classes. Time-variant attributes should go in one or more time-variant volatility classes. Notice that attributes of a single volatility class are supposed to have a common lifespan history (they are assumed to change together over time).

For the Movie entity in the MovieDB sample, three volatility classes have been identified (see Figure 87 on page 148): One is the time-invariant class, containing the MovieID (the key) and the Title and YearOfRelease attributes (which are not supposed to change over time). For Budget values and TicketSales, two separate classes are set up because both values are assumed to have different patterns of change over time.

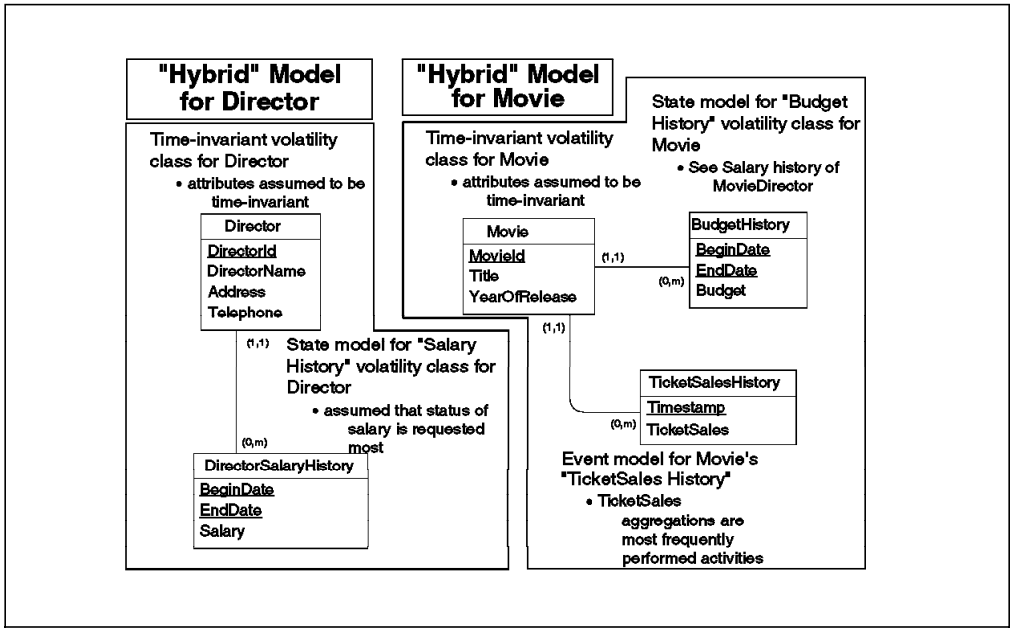


Figure 87. Director and Movie Volatility Classes.

Different volatility classes can be given different time stamps for recording history records which are best suited for the purpose. For instance, Budget is assumed to change infrequently and is therefore recorded using a state model with a variable-duration interval time stamp. TicketSales, however, is expected to bring in a new value each time a new record comes in. We have set up a model for the TicketSales history that expects new TicketSales values to be delivered to the warehouse each day.

Adding Entities for Transactions and Events: One of the typical characteristics of a data warehouse is that it contains information about business transactions and events. Transactions and events are of significant importance for many information analysis processes. They can be used to group history records together (for instance, all sales records that belong to a single customer sale, including the mode of payment used by the customer for the sales transaction) or provide the analyst a treasure of additional information to explain what happened or why things happened.

The pure model transformation process has a serious flaw: It will not support transactions and events that are not present in the original data model (a model transformation process obviously does not invent new things). It is therefore advisable to consider adding other entities to the historical model, primarily to be able to capture business transactions and events.

Adding Relationships and the History of Relationship Changes Relationships in the original data model can be handled in a way that is similar to our way of handling entities. For each of the relationships in the original model, we suggest investigating whether a history of the relationship itself should be kept or not.

If a history of changes should not be captured for a particular relationship, the original relationship can simply be registered in the historical model as a relationship between the time-invariant volatility classes (the anchor records). If a history must be captured, the relationship must be turned into an association between the original entities, and time stamps should be added as attributes of the resulting association (see Figure 88 on page 149).

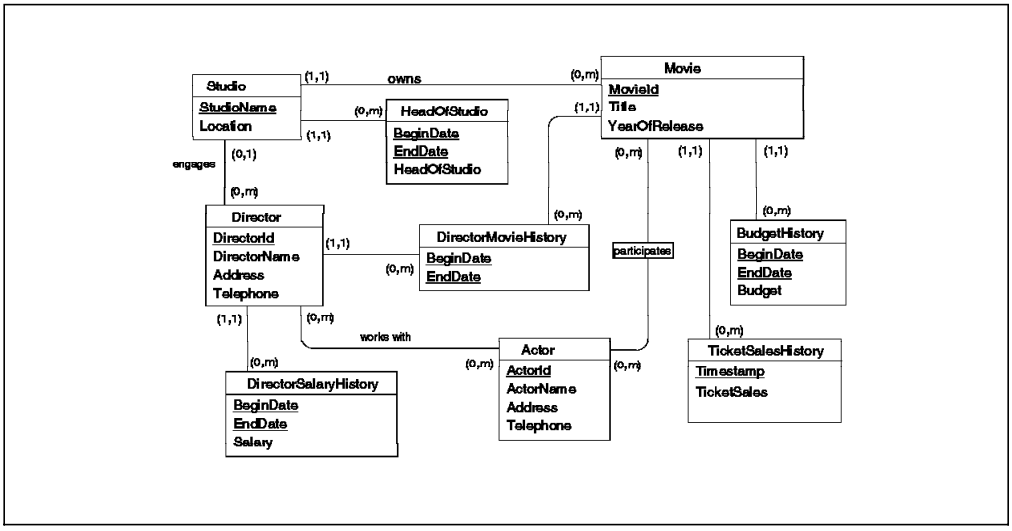


Figure 88. Temporal Model for MovieDB.

Grouping Time-Variant Classes of Attributes: Temporal modeling using the model transformation techniques presented above results in the creation of several history records. To reduce the complexity of the resulting model, some of the history records may be grouped together into larger construct (see Figure 89).

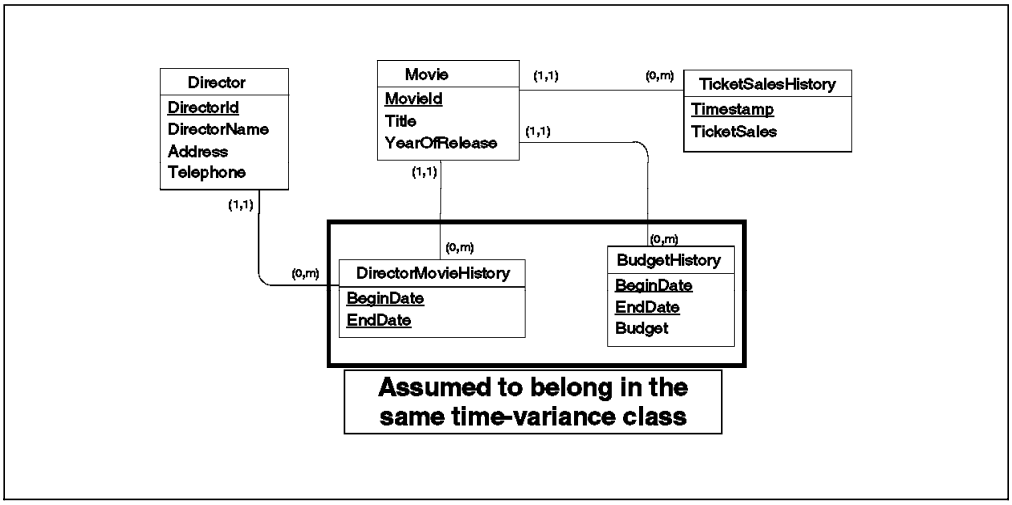


Figure 89. Grouping of Time-Variant Classes of Attributes.

Notice that as a consequence of applying this grouping technique, redundancy is introduced in the model.

Advanced Temporal Modeling Techniques: To conclude the coverage of temporal modeling, we present below a survey of some of the most important advanced temporal modeling techniques for data warehouses.

Adding Temporal Constraints to a Model: Constraints are important elements of a conceptual model. Some constraints are captured in the model as structural relationships between entities. Other constraints have to be specified explicitly through a constraint specification language.

Usually, the application developer uses constraints to implement the code for the constraint in the application services. Data warehouses obviously do not have such application logic. Here, the context is different.

Temporal constraints express a constraint between elements of a temporal model. The constraint is temporal if it includes time-related aspects. As an example (using the MovieDB model), a temporal constraint could express that two directors cannot direct a given movie at the same time.

Although some temporal constraints could be coded in the model through structural relationships, most temporal constraints quickly become too complex for this. Temporal models therefore often are extended with temporal constraints expressed through a functional language. These constraints are important elements to consider when populating the data warehouse with newly arriving data records or data change records. Temporal constraints are also important for end users, as part of the metadata, because they can further help end users in their information analysis activities.

Modeling Lifespan Histories of Database Objects: Knowing the lifespan history of a database object can contribute considerably to the consistency of the data in the data warehouse. It can also be of considerable help for end users in performing their data analysis activities.

The lifespan history of a database object is a representation of the representative activities that take place during the lifespan of an object. Typical activities that are of interest to a data warehouse modeler are creation, change, and deletion of the object. These activities obviously only form the basis of any lifespan history. There are certainly other situations in which lifespan history modeling may provide help for solving problems; for example, situations involving changes to the keys in the operational databases.

Lifespan history modeling can easily evolve into an elaborate technique. In its broad context, the technique is used to study the evolution over time of a given database object. The results of this technique can be used to refine temporal models.

Modeling Time-Variancy at the Schema Level: Modeling time-variancy at the schema level is concerned with building models for the data warehouse that can sustain adding or dropping attributes from the model, over time. We have seen that the technique is required in some cases where hierarchies within slow-varying dimensions are changed. The techniques whereby separate history records are created for various volatility classes can be extended to provide support for modeling schema level changes.

Some Conclusions: We have surveyed and illustrated several basic and advanced techniques for temporal data modeling for the data warehouse. The area of temporal modeling is quite complex, and few if any complete modeling approaches are available. Modeling tools also provide only limited support for temporal modeling. We advocate that temporal modeling be applied with a serious touch of reality. Not only can the techniques be complex, the resulting models also tend to become complex, especially when end users have to deal with the models directly.

If such modeling techniques may turn out to be required for a particular project, complexity for end users can be reduced by building a front-end data model for

the end users that involves simpler temporal constructs, for example, a model that uses the cumulative snapshot approach.

8.4.4.5 Selecting a Data Warehouse Modeling Approach

Dimensional modeling is a very powerful data warehouse modeling technique. According to Ralph Kimball, ″dimensional modeling is the only viable technique for databases that are designed to support end-user queries in a data warehouse″ (Ralph Kimball, ″A Dimensional Modeling Manifesto,″ *DBMS Online*, August 1997).

Dimensional modeling is different from ER modeling in that it focuses on presenting data to end users much more intuitively and with a special emphasis on supporting high-performance querying of potentially very large fact tables.

As we have illustrated in this chapter, star models play a particular role in the context of dimensional modeling. Star models are very condensed and packed representations of the dimensions used by end users in their query and analysis activities. Because of their compactness, authors like Ralph Kimball consider star models roughly speaking as the only good representations for supporting end user information analysis processing.

Reasons for this are twofold. First, because star models represent the dimensions as flat tables, end users are not confronted with complex dimension structures in the form of normalized ER schemas. This gives them the opportunity to work with the dimensional model without having to learn how to interpret (complex) ER schemas correctly.

Second, because star schemas are flattened structures, end users and/or the end-user tools do not have to join entities or tables in the dimensions to find and use related data items for selecting and aggregating measures and facts. This results in enhanced query performance.

Star modeling approaches therefore are considered by some to be the modeling approaches that both simplify and improve the performance of end-user queries.

Snowflake models tend to offer more support for complex or more elaborate data modeling problems. When a dimension incorporates several different parallel aggregation paths, star models have difficulty in expressing reality in a concise way. Also, when dimensions are varying in time, the slow varying temporal aspects of dimensions cannot really nicely be represented by the flat dimension tables of star schemas. There are several other modeling situations, all involving elaborate modeling problems, where one can indeed recognize that snowflake models are more appropriate than stars. (We do not imply that stars cannot be used for these modeling situations; rather, stars do not directly capture the essence of the problems.)

In the data warehouse modeling arena, practitioners are more and more often confronted with a (fiercely) raging debate about how exactly a data warehouse should be modeled. Although we would like to stay out of this debate as much as possible, we cannot end this chapter without an attempt to provide a synthesis consisting of a set of guidelines to set up a data warehouse modeling approach that is suitable for medium- to large-scale data warehouse and data mart modeling.

Our position is that there is room for any of the modeling approaches, be it dimensional modeling using star or snowflake models or ER modeling involving

temporal modeling for producing the historical data model for the data warehouse. Successful data warehouse modeling requires that the techniques be applied in the right context and within the bounds of their most suitable areas of applicability.

Considerations for ER Modeling: ER modeling is the most popular approach for modeling OLTP databases. In this area, it is challenged only by object-oriented modeling approaches. ER modeling is also the de facto standard approach for OLTP database model reengineering and database model integration. This implies that ER modeling is the most suitable modeling approach for enabling operational data stores. For the same reason, for producing source-driven data warehouse models, ER modeling is the most suitable approach.

Despite the negative comments that dimensional modeling advocates have made against ER modeling, it remains a viable approach for data warehouses and for data marts. Because ER models can be very hard to read and interpret, however, it should be expected that end users will not be skilled in interpreting complex (possibly wall-sized) models of reality. ER models usually are too technical for end users to work with, directly.

Considerations for Dimensional Modeling: It should be clear by now that dimensional models can indeed provide very straightforward-looking representations of the data used by end users in their information analysis activities. This is obviously not sheer magic, because the modeling approach was conceived for just that purpose.

Dimensional modeling can be recommended as the modeling approach to use, in the following situations:

1. When end-user requirements are analyzed and validated, because end users are heavily involved in this process, the approach based on producing the initial dimensional models described in this chapter offers an opportunity to combine simplicity of the resulting model diagrams and power of the modeling capabilities. Experience has shown that the approach is usable with direct end-user involvement.
2. When data models are produced for data marts that will be implemented with OLAP tools that are based on the multidimensional data modeling paradigm.
3. When data warehouse models are produced for end-user queryable databases. In this case, we recommend producing flattened star models, unless the star modeling approach would lead to modeling solutions that do not correspond to reality. End-user models should not simply be simple, they should be intuitive and represent directly the perception of reality that end users (who should be considered experts in their own business domain) have.

Two-Tiered Data Modeling: Careful readers should have noticed that the above mentioned guidelines do not imply that dimensional modeling is the one and only approach for modeling data marts or that we proclaim star schemas as the only possible or acceptable solutions for data marts.

Modeling a data mart with a broad scope of information coverage will almost certainly require the use of different modeling approaches, because such marts tend to have complex data models. For such projects, we propose using intuitive star schemas for those parts of the data mart end users directly interact with

(the ″end-user queryable databases″ within the data mart). The core model of the data mart will probably involve an ER model or an elaborated snowflake model.

We call such an approach a ″two-tiered data modeling approach,″ because it results in a single, consistent data model that consists of two apparent ″layers″ in the model: one the end users are working with and one that represents the core of the data mart. Both layers of a two-tiered data warehouse model must be mapped.

Despite the inherent complexities of two-tiered data modeling, we do recommend its use for broad scope data marts and for data warehouses.

Dimensional Modeling Supporting Drill Across: In his book, *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses*, Ralph Kimball illustrates yet another technique for building dimensional models that span across subject areas. The technique relies on the principle of an operation called ″drilling across″ dimensions that are common to multiple dimensional models.

Drill across requires that dimensions that span the dimensional models (and have the same meaning) are physically identical. Kimball states that ″in order to support drill-across applications, all constraints on dimension attributes must evaluate to exactly the same set of dimensional entities from one database to the next″ (Ralph Kimball, *The Data Warehouse Toolkit*, p. 84). If a given dimension is common to two dimensional models and the dimension has an identical layout in both models, the requirement quoted above obviously is satisfied.

The drill-across dimensional modeling principle can be used for producing models that enable end users to analyze several facts possibly belonging to different subject areas, as long as the facts share their common dimensions. This obviously implies that producing shareable, generic dimensional models is a key part of the approach.

Shared, multidimensional data models can thus be produced as federations of dimensional models, where each part of the federated model represents a particular part of the business process.

This approach extends standard dimensional modeling beyond the individual data analysis processing performed by identifiable groups of end users, but only if the dimensions in the models are made conformant and can be consolidated.

If, in addition, facts in a dimensional model are recorded at low levels of granularities, the resulting dimensional models are a very powerful and flexible representation of the reality end users are querying and analyzing.

This modeling approach is also suitable for an architected approach based on the ″dependent data mart″ principle (see Chapter 4, “Data Warehousing Architecture and Implementation Choices” on page 15).

Modeling Corporate Historical Databases: A data warehouse is by definition a historical or temporal database. Although OLTP databases may, in exceptional cases, contain parts that capture history, there is a significant difference between the two, and this difference stems from the difference in historical perspective: OLTP databases rarely capture more than a few weeks or months of

history, whereas data warehouses should be able to capture 3 to 5 to even 10 years of history, basically for all of the data that is recorded in the data warehouse.

In one sense, a historical database is a dimensional database, the dimension being time. In that sense, a historical data model could be developed using a dimensional modeling approach. In the context of corporate data warehouse modeling, building the backbone of a large-scale data warehouse, we believe this makes no sense. In this case, the recommended approach is an ER modeling approach that is extended with time variancy or temporal modeling techniques, as described earlier in this chapter.

There are two basic reasons for the above-mentioned recommendation:

- Corporate historical models most often emerge from an inside-out approach, using existing OLTP models as the starting point of the modeling process. In such cases, reengineering existing source data models and integrating them are vital processes. Adding time to the integrated source data model can then be considered a model transformation process; suitable techniques for doing this have been described in various sections of this chapter.
- Historical data models can become quite complicated. In some cases, they are inherently unintuitive for end users anyway. In this case, one of the basic premises for using dimensional modeling simply disappears.

Notice that this observation implies that end-users will find it difficult to query such historical or temporal models. The complications of a historical data model will therefore have to be hidden from end users, using tools or two-tiered data modeling or an application layer.

A modeling approach for building corporate historical data models basically consists of two major steps. The first step is to consolidate (existing) source data models into a single unified model. The second step is to add the time dimension to the consolidated model, very much according to the techniques described in 8.4.4.4, “Temporal Data Modeling” on page 139.

In data warehousing, the whole process of constructing a corporate historical data model must take place against the background of a corporate data architecture or enterprise data model. The data architecture must provide the framework to enhance consistency of the outcome of the modeling process. The corporate data architecture should also maximize scalability and extensibility of the historical model. The role of the data architect in this process obviously is of vital importance.

Chapter 9. Selecting a Modeling Tool

Modeling for data warehousing is significantly different from modeling for operational systems. In data warehousing, quality and content are more important than retrieval response time. Structure and understanding of the data, for access and analysis, by business users is a base criterion in modeling for data warehousing, whereas operational systems are more oriented toward use by software specialists for creation of applications. Data warehousing also is more concerned with data transformation, aggregation, subsetting, controlling, and other process-oriented tasks that are typically not of concern in an operational system. The data warehouse data model also requires information about both the source data that will be used as input and how that data will be transformed and flow to the target data warehouse databases. Thus, the functions required for data modeling tools for data warehousing data modeling have significantly different requirements from those required for traditional data modeling for operational systems.

In this chapter we outline some of the functions that are of importance for data modeling tools to support modeling for a data warehouse. The key functions we cover are: diagram notation for both ER models and dimensional models, reverse engineering, forward engineering, source to target mapping of data, data dictionary, and reporting. We conclude with a list of modeling tools.

9.1 Diagram Notation

Both ER modeling and dimensional modeling notation must be available in the data modeling tool. Most models for operational systems databases were built with an ER modeling tool. Clearly, any modeling tool must, at a minimum, support ER modeling notation. This is very important even for functions that are not related to data warehousing, such as reverse engineering. In addition, it may be desirable to extend, or aggregate, any existing data models to move toward an enterprise data model. Although not a requirement, such a model could be very useful as the starting point for developing the data warehouse data model.

As discussed throughout this book, more and more data warehouse database designs are incorporating dimensional modeling techniques. To be effective as a data modeling tool for data warehouse modeling, a tool must support the design of dimensional models.

9.1.1 ER Modeling

ER modeling notation supports entities and relationships. Relationships have a degree and a cardinality, and they can also have attributes and constraints. The number of different entities that participate in a relationship determines its degree. The cardinality of a relationship specifies the number of occurrences of one entity that can or must be associated with each occurrence of another entity. Each relationship has a minimum and maximum cardinality in each direction of the relationship. An attribute is a characteristic of an entity. A constraint is a rule that governs the validity of the data manipulation operations, such as insert, delete, and update, associated with an entity or relationship.

The data modeling tool should actually support several of the ER modeling notations such as Chen, Integration Definition for Information Modeling,

Information Engineering, and Zachman notation. An efficient and effective data modeling tool will enable you to create the data model in one notation and convert it to another notation without losing the meaning of the model.

9.1.2 Dimensional Modeling

Dimensional modeling notation must support both the star and snowflake model variations. Because both model variations are concerned with fact tables and dimension tables, the notation must be able to distinguish between them. For example, a color or special symbol could be used to distinguish the fact tables from the dimensional tables. A robust data modeling tool would also support notation for aggregation, as this is a key function in data warehousing. Even though it may not be as critical as with operational systems, performance is always an issue. Indexes have the greatest impact on performance, so the tool must support the creation of keys for the tables and the selection of indexes.

9.2 Reverse Engineering

Reverse engineering is the creation of a model based on the source data in the operational environment as well as from other external sources of data. Those sources could include relational and nonrelational databases as well as other types of file-oriented systems. Other sources of data would include indexed files and flat files, as well as operational systems sources, such as COBOL copy books and PL/1 libraries. The reverse engineered model may be used as the basis for the data warehouse model or simply for information about the data structure of the source data.

A good data warehouse data modeling tool is one that enables you to use reverse engineering to keep the model synchronized with the target database. Often the database administrator or a developer will make changes to the database instead of the model because of time. When changes are made to the target database, they are reflected in the data model through the modeling tool.

9.3 Forward Engineering

Forward engineering is the creation of the data definition language (DDL) for the target tables in the data warehouse databases. The tool should be capable of supporting both relational and multidimensional databases. At a minimum, clearly, the tool must support the structure of the database management system being used for the target data warehouse. It must be capable of generating the DDL for the databases in that target data warehouse. The DDL should support creation of the tables, views, indexes, primary keys, foreign keys, triggers, stored procedures, table spaces, and storage groups.

The tool being used to create the data warehouse must enable you to execute the DDL automatically in the target database or to save the DDL to a script file. However, if the DDL is at least saved to a script file, you can then manually run it. Support must include the capability to either generate the complete database or incrementally generate parts of the database.

9.4 Source to Target Mapping

Source to target mapping is the linking of source data in the operational systems and external sources to the data in the databases in the target data warehouse. The data modeling tool must enable you to specify where the data for the data warehouse originates and the processing tasks required to transform the data for the data warehouse environment. A good data modeling tool will use the source to target mapping to generate scripts to be used by external programs, or SQL, for the data transformation.

9.5 Data Dictionary (Repository)

The data dictionary, or repository, contains the metadata that describes the data model. It is this metadata that contains all the information about the data sources, target data warehouse databases, and all the processes required to cleanse, transform, aggregate, and maintain the environment.

A powerful data modeling tool would include the following information about the data in the model:

- Model names
- Model definition
- Model purpose
- Dimension names
- Dimension aliases
- Dimension definitions
- Dimension attribute names
- Dimension attribute aliases
- Dimension attribute definitions
- Dimension attribute data type
- Dimension attribute domain
- Dimension attribute derivation rules
- Fact names
- Fact aliases
- Fact definitions
- Measure names
- Measure aliases
- Measure definitions
- Measure data type
- Measure domain
- Measure derivation rules
- Dimension hierarchy data
- Dimension change rule data
- Dimension load frequency data
- Relationships among the dimensions and facts
- Business use of the data
- Applications that use the data
- Owner of the data
- Structure of data including size and data type
- Physical location of data
- Business rules

9.6 Reporting

Reporting is an important function of the data modeling tool and should include reports on:

- Fact and dimension tables
- Specific facts and attributes in the fact and dimension tables
- Primary and foreign keys
- Indexes
- Metadata
- Statistics about the model
- Errors that exist in the model

9.7 Tools

The following is a partial list of some of the tools available in the marketplace at the time this redbook was written. The presence of a tool in the list does not imply that it is recommended or has all of the required capabilities. Use the list as a starting point in your search for an appropriate data warehouse data modeling tool.

- CAST DB-Builder (www.castsoftware.com)
- Cayenne Terrain (www.cayennesoft.com)
- Embarcadero Technologies ER/Studio (www.embarcadero.com)
- IBM VisualAge DataAtlas (www.software.ibm.com)
- Intersolv Excelerator II (www.intersolv.com)
- Logic Works ERwin (www.logicworks.com)
- Popkin System Architect (www.popkin.com)
- Powersoft PowerDesigner WarehouseArchitect (www.powersoft.com)
- Sterling ADW (www.sterling.com)

Chapter 10. Populating the Data Warehouse

Populating is the process of getting the source data from operational and external systems into the data warehouse and data marts (see Figure 90). The data is captured from the operational and external systems, transformed into a usable format for the data warehouse, and finally loaded into the data warehouse or the data mart. Populating can affect the data model, and the data model can affect the populating process.

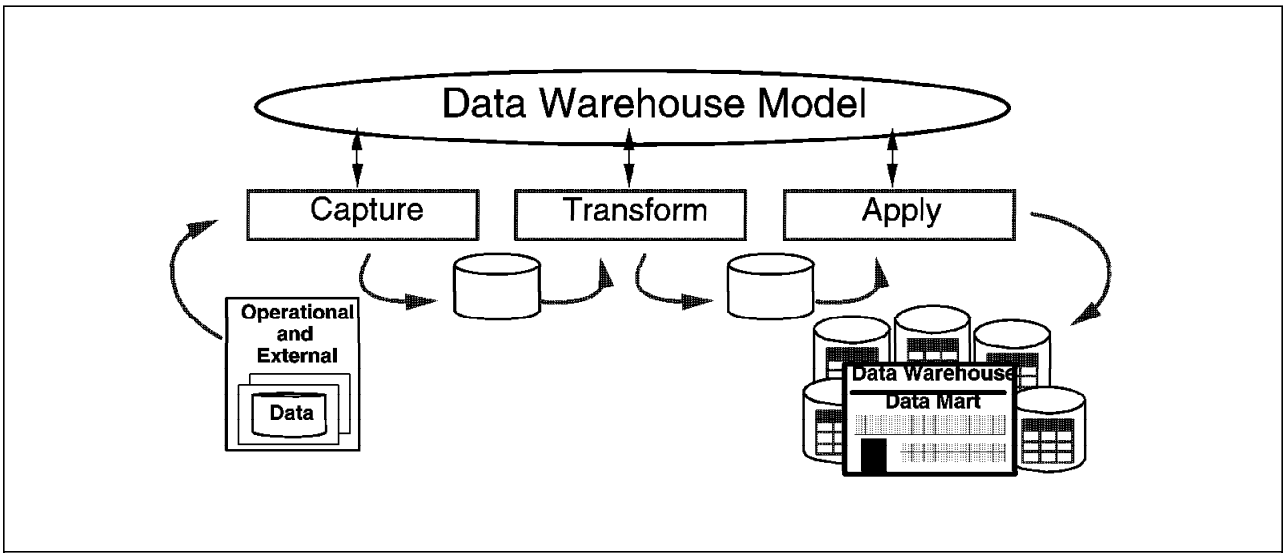


Figure 90. Populating the Data Warehouse.

10.1 Capture

Capture is the process of collecting the source data from the operational systems and other external sources. The source of data for the capture process includes file formats and both relational and nonrelational database management systems. The data can be captured from many types of files, including extract files or tables, image copies, changed data files or tables, DBMS logs or journals, message files, and event logs. The type of capture file depends on the technique used for capturing the data. Data capturing techniques include source data extraction, DBMS log capture, triggered capture, application-assisted capture, time-stamp-based capture, and file comparison capture (see Table 3 on page 160).

Source data extraction provides a static snapshot of source data as of a specific point in time. It is sufficient to support a temporal data model that does not have a requirement for a continuous history. Source data extraction can produce extract files, tables, or image copies.

Log capture enables the data to be captured from the DBMS logging system. It has minimal impact on the database or the operational systems that are accessing the database. This technique does require a clear understanding of the format of the log records and fairly sophisticated programming to extract only the data of interest.

Triggers are procedures, supported by most database management systems, that provide for the execution of SQL or complex applications on the basis of recognition of a specific event in the database. These triggers can enable any type of capture. The trigger itself simply recognizes the event and invokes the procedure. It is up to the user to actually develop, test, and maintain the procedure. This technique must be used with care because it is controlled more by the people writing the procedures rather than by the database management system. Therefore, it is open to easy access and changes as well as interference by other triggering mechanisms.

Application-assisted capture involves programming logic in existing operational system applications. This implies total control by the application programmer along with all the responsibilities for testing and maintenance. Although a valid technique, it is considered better to have application-assisted capture performed by products developed specifically for this purpose, rather than to develop your own customized application.

DBMS log capture, triggered capture, and application-assisted capture can produce an incremental record of source changes, to enable use of a continuous history model. Each of these techniques typically requires some other facility for the initial load of data.

Time-stamp-based capture is a simple technique that involves checking a time stamp value to determine whether the record has changed since the last capture. If a record has changed, or a new record has been added, it is captured to a file or table for subsequent processing.

A technique that has been used for many years is file comparison. Although it may not be as efficient, it is an easy technique to understand and implement. It involves saving a snapshot of the data source at a specific point in time of data capture. At a later point in time, the current file is compared with the previous snapshot. Any changes and additions that are detected are captured to a separate file for subsequent processing and adding to the data warehouse databases. Time-stamp-based capture, with its file comparison technique, produces a record of the incremental changes that enables support of a continuous history model. However, care must be exercised because all changes to the operational data may not have been recorded. Changes can get lost because more than one change of a record may occur between capture points. Therefore, the history captured would be based on points in time rather than a record of the continuous change history.

Table 3. Capture Techniques

Technique	**Initial Load**	**Incremental Load - Each Change**	**Incremental Load - Periodic Change**
Source data extraction	X		
DBMS log capture		X	
Triggered capture		X	
Application-assisted capture		X	
Time-stamp-based capture			X
File comparison capture			X

10.2 Transform

The transform process converts the captured source data into a format and structure suitable for loading into the data warehouse. The mapping characteristics used to transform the source data are captured and stored as metadata. This defines any changes that are required prior to loading the data into the data warehouse. This process will help to resolve the anomalies in the source data and produce a high quality data source for the target data warehouse. Transformation of data can occur at the record level or at the attribute level. The basic techniques include structural transformation, content transformation, and functional transformation.

Structural transformation changes the structure of the source records to that of the target database. This technique transforms data at the record level. These transformations occur by selecting only a subset of records from the source records, by selecting a subset of records from the source records and mapping to different target records, by selecting a subset of different records from the source records and mapping to the same target record, or by some combination of each. If a fact table in the model holds data based on events, records should be created only when the event occurs. However, if a fact table holds data based on the state of the data, each time the data is captured a record should be created for the target table.

Content transformation changes data values in the records. This technique transforms data at the attribute level. Content transformation converts values by use of algorithms or by use of data transformation tables.

Functional transformation creates new data values in the target records based on data in the source records. This technique transforms data at the attribute level. These transformations occur either through data aggregation or enrichment. Aggregation is the calculation of derived values such as totals and averages based on multiple attributes in different records. Enrichment combines two or more data values and creates one or more new attributes from a single source record or multiple source records that can be from the same or different sources.

The transformation process may require processing through the captured data several times because the data may be used to populate various records during the apply process. Data values may be used in a fact table as a measure and they may also be used to calculate aggregations. This may require going through the source records more than once. The first pass would be to create records for the fact table and the second to create records for the aggregations.

10.3 Apply

The apply process uses the files or tables created in the transform process and applies them to the relevant data warehouse or data mart.

There are four basic techniques for applying data: load, append, constructive merge, and destructive merge. Load replaces the existing data in the target data warehouse tables with that created in the transform process. If the target tables do not exist, the load process can create the table. Append loads new data from the transform file or table to an already existing table by appending the new data to the end of the existing data. Constructive merge appends the new records to the existing target table and updates an end time value in the

record whose state is being superseded. Destructive merge overwrites existing records with new data.

10.4 Importance to Modeling

When the data warehouse model is being created, consideration must be given to the plan for populating the data warehouse. Limitations in the operational system data and processes can affect the data availability and quality. In addition, the populating process requires that the data model be examined because it is the blueprint for the data warehouse. The modeling process and the populating process affect each other.

The data warehouse model determines what source data will be needed, the format of the data, and the time interval of data capture activity. If the data required is not available in the operational system, it will have to be created. For example, sources of existing data may have to be calculated to create a required new data element. In the case study, the Sale fact requires Total Cost and Total Revenue. However, these values do not reside in the source data model. Therefore, Total Cost and Total Revenue must be calculated. In this case, Total Cost is calculated by adding the cost of each component, and Total Revenue is calculated by adding all of the Order Line′s Negotiated Unit Selling Price times Quantity Ordered. The model may also affect the transform process. For example, the data may need to be processed more than once to create all the necessary records for the data warehouse.

The populating process may also influence the data warehouse model. When data is not available or is costly to retrieve, it may have to be removed from the model. Or, the timeliness of the data may have to change because of physical constraints of the operational system, which will affect the time dimension in the model. For example, in the case study, the Time dimension contains three types of dates: Date, Week of Year, and Month of Year. If populating can occur only on a weekly basis because of technology reasons, the granularity of the Time dimension would have to be changed, and the Date attribute would have to be removed.

Appendix A. The CelDial Case Study

Before reviewing this case study, you should be familiar with the material presented in Chapter 7, “The Process of Data Warehousing” on page 49 from the beginning to the end of 7.3, “Requirements Gathering” on page 51. The case study is designed to enable you to:

- Understand the information presented in a dimensional data model
- Create a dimensional data model based on a given set of business requirements
- Define and document the process of extracting and transforming data from a given set of sources and populating the target data warehouse

We begin with a definition of a fictional company, CelDial, and the presentation of a business problem to be solved. We then define our data warehouse project and the business needs on which it is based. An ER model of the source data is provided as a starting point. We close the case study with a proposed solution consisting of a dimensional model and the supporting metadata.

Please review the case study up to but not including the proposed solution. Then return to 7.4, “Modeling the Data Warehouse” on page 53 where we document the development of the solution. We include the solution in this appendix for completeness only.

A.1 CelDial - The Company

CelDial Corporation started as a manufacturer of cellular telephones. It quickly expanded to include a broad range of telecommunication products. As the demand for, and size of, its suite of products grew, CelDial closed down distribution channels and opened its own sales outlets.

In the past year CelDial opened new plants, 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.

CelDial′s growth has started to level off, and 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 manufacturing plant or 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 warehouse. After consideration of the potential costs and benefits, management agreed.

A.2 Project Definition

Senior management and IT put together a project definition consisting of the following objective and scope:

Project Objective

To create a data warehouse to facilitate the analysis of cost and revenue data for products manufactured and sold by CelDial.

Project Scope

The project shall be limited to direct costs and revenues associated with products. Currently, CelDial's manufacturing costs cannot be allocated at the product level. Therefore, only component costs can be included. At a future time, rules for allocation of manufacturing and overhead costs may be created, so the data warehouse should be flexible enough to accommodate future changes.

IT created a team consisting of one data analyst, one process analyst, one manufacturing plant manager, and one sales region manager for the project.

A.3 Defining the Business Need

First, the project team defined what they needed to investigate in order to understand the business need. To that end, the team identified the following areas of interest:

- Life cycle of a product
- Anatomy of a sale
- Structure of the organization
- Defining cost and revenue
- What do the users want?

A.3.1 Life Cycle of a Product

The project team first studied the life cycle of a product. Each manufacturing plant has a research group that tests new product ideas. Only after the manufacturing process has been completely defined and approval for the new product has been obtained is the product information added to the company's records. Once the product information is complete, all manufacturing plants can produce it.

A product has a base set of common components. Additional components are added to the base set to create specific models of the product. Currently, CelDial has 300 models of products. This number is fairly constant as the rate of new models being created approximately equals the rate of old models being discontinued. Approximately 10 models per week experience a cost or price change. For each model of each product, a decision is made about whether or not it is eligible for discounting. When a model is deemed eligible for discounting, the salesperson may discount the price if the customer buys a large quantity of the model or a combination of models. In a retail store (see A.3.2, "Anatomy of a Sale" on page 165) the store manager must approve such a discount.

The plant keeps an inventory of product models. When the quantity on hand for a model falls below a predetermined level, a work order is created to cause more of the model to be manufactured. Once a model is manufactured, it is stored at the manufacturing plant until it is requested by a sales outlet.

The sales outlet is responsible for selling the model. When a decision is made to stop making a model, data about the model is kept on file for six months after the last unit of the model has been sold or discarded. Data about a product is removed at the same time as data about the last model for the product is removed.

A.3.2 Anatomy of a Sale

There are two types of sales outlets: corporate sales office and retail store. 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 CelDial's 30 sales representatives is assigned to each corporate customer. CelDial currently serves 3000 corporate customers. 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 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 10 product models.

A retail store sells over the counter. Unless a discount is negotiated, the suggested retail price is charged. 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 1000 orders per day, seven days per week. Each order consists of an average of two product models.

A.3.3 Structure of the Organization

It was clear to the team that understanding products and sales was not enough; an understanding of the organization was also necessary. The regional sales manager provided an up-to-date copy of the organization structure (see Figure 91 on page 166).

A.3.4 Defining Cost and Revenue

The project team must clearly define cost and revenue in order for users to effectively analyze those factors.

For each product model, the cost of each component is multiplied by the number of components used to manufacture the model. The sum of results for all components that make up the model is the cost of that model.

For each product model, the negotiated unit selling price is multiplied by the quantity sold. The sum of results for all order lines that sell the model is the revenue for that model.

When trying to relate the cost of a model to its revenue, the team discovered that once a model was manufactured and added to the quantity on hand in inventory, the cost of that unit of the model could not be definitively identified. Even though the cost of a component is kept, it is only used to calculate a current value of the inventory. Actual cost is recorded only in the company's financial system, with no reference to the quantity manufactured.

The result of this determination was twofold. First, the team requested that the operational systems be changed to start recording the actual cost of a manufactured model. However, both management and the project team recognized that this was a significant change, and that waiting for it would

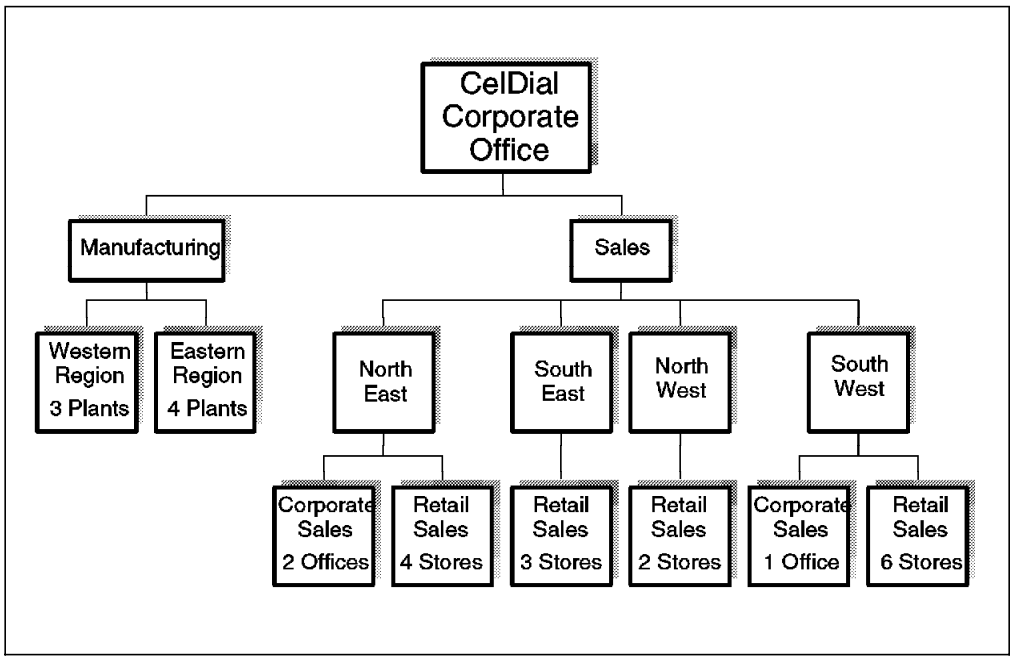


Figure 91. CelDial Organization Chart

severely impact the progress of the project. Therefore, and based on the fact that component costs changed infrequently and by small amounts, the team defined this rule: The revenue from the sale of a model is always recorded with the current unit cost of the model, regardless of the cost of the model at the time it was manufactured.

A.3.5 What Do the Users Want?

Because the objective of the project was to create a collection of data that users could effectively analyze, the project team decided to identify a set of typical questions users wanted the data to answer. Clearly, this would not be an exhaustive list. The answer to one question would certainly determine what the next question, if any, might be. As well, one purpose of the data warehouse is to allow the asking of as yet unknown questions. If users simply want to answer a rigid set of questions, creating a set of reports would likely fill the need. With this in mind, the team defined the following set of questions:

1. What is the average quantity on hand and reorder level this month for each model in each manufacturing plant?
2. What are the total cost and revenue for each model sold today, summarized by outlet, outlet type, region, and corporate sales levels?
3. What are the total cost and revenue for each model sold today, summarized by manufacturing plant and region?
4. What percentage of models are eligible for discounting, and of those, what percentage are actually discounted when sold, by store, for all sales this week? This month?
5. For each model sold this month, what is the percentage sold retail, the percentage sold corporately through an order desk, and the percentage sold corporately by a salesperson?
6. Which models and products have not sold in the last week? The last month?

7. What are the top five models sold last month by total revenue? By quantity sold? By total cost?
8. Which sales outlets had no sales recorded last month for each of the models in each of the three top five lists?
9. Which salespersons had no sales recorded last month for each of the models in each of the three top five lists?

As well as being able to answer the above questions, the users want to be able to review up to three complete years of data to analyze how the answers to these questions change over time.

A.4 Getting the Data

Once the team defined the business need, the next step was to find the data necessary to build a data warehouse that would support that business need. To that end, the data analyst provided the ER model for all relevant data available in the current operational systems (see Figure 92 on page 168).

As well, the data analyst provided the record layouts for two change transaction logs: one for products and models and one for components and product components.

The layout for product and model changes is:

Name	*Data Type*	*Length*	*Start Position*
Product ID	Numeric	5	1
Model ID	Numeric	5	6
Product Description	Character	40	11
Suggested Wholesale Price	Numeric(9,2)	5	51
Suggested Retail Price	Numeric(9,2)	5	56
Eligible for Volume Discount	Character	1	57

The layout for component and product component changes is:

Name	*Data Type*	*Length*	*Start Position*
Component ID	Numeric	5	1
Product ID	Numeric	5	6
Model ID	Numeric	5	11
Component Description	Character	40	16
Unit Cost	Numeric(9,2)	5	56
Number of Components	Numeric	5	61

A.5 CelDial Dimensional Models - Proposed Solution

We believe that CelDial's needs can be met by creating and implementing two dimensional models. We include them here for your consideration (see Figure 93 on page 169 and Figure 94 on page 170).

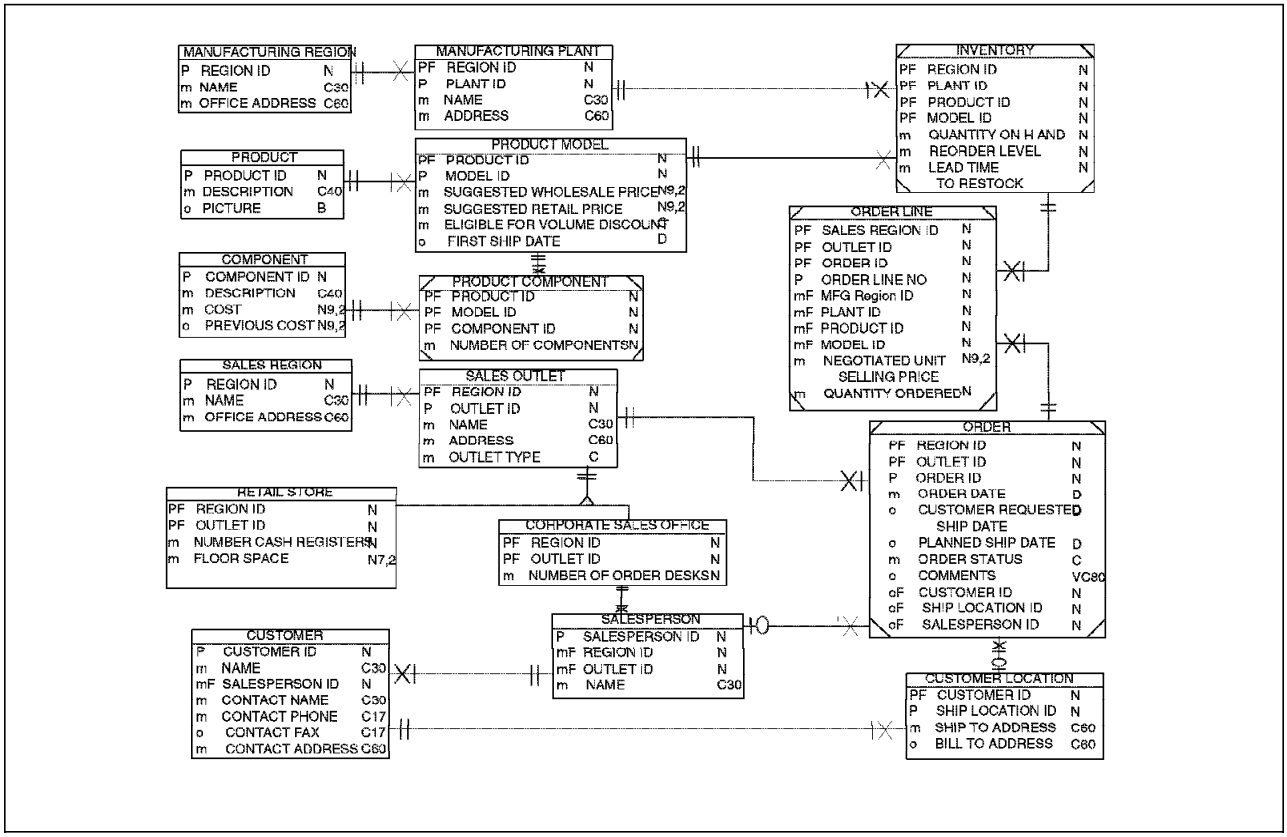


Figure 92. Subset of CelDial Corporate ER Model

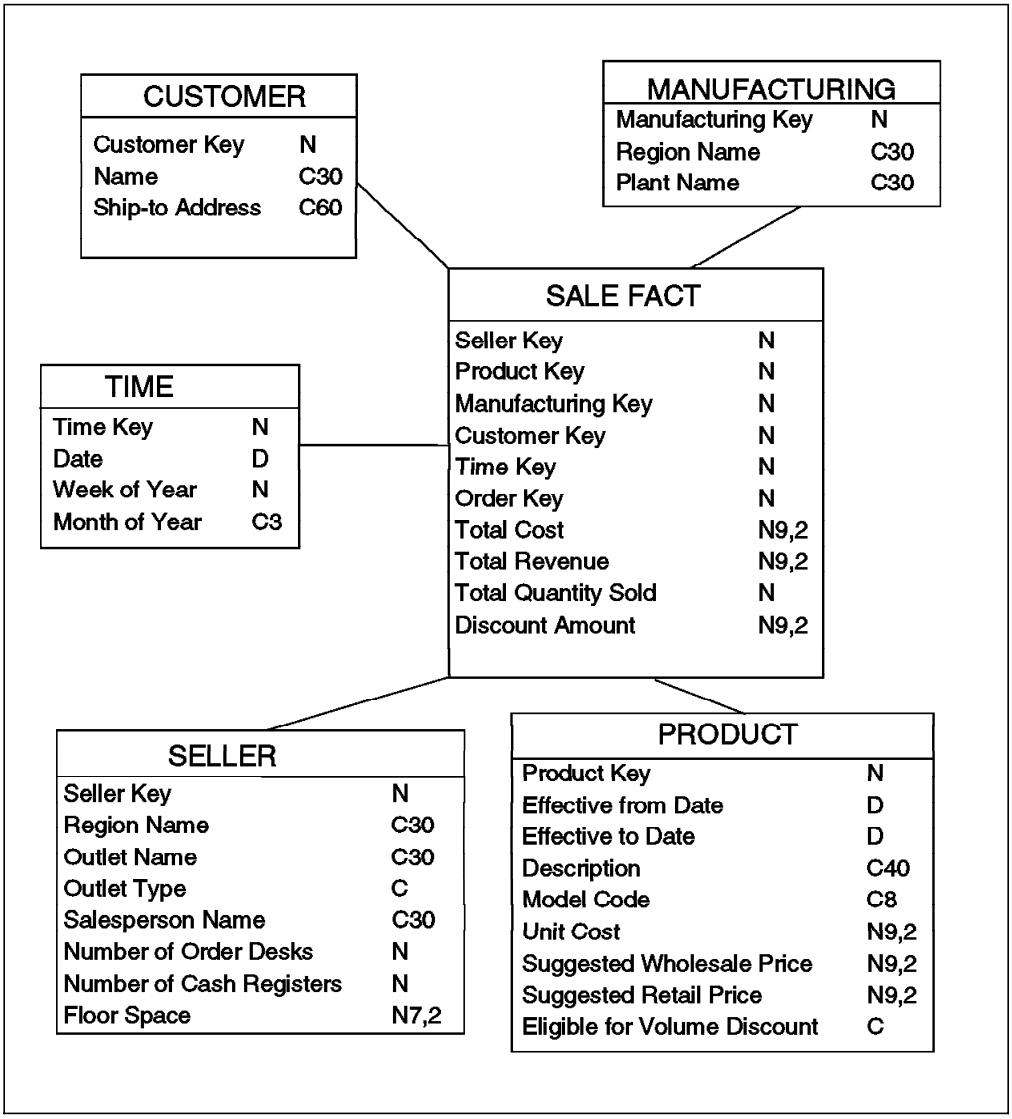


Figure 93. Dimensional Model for CelDial Product Sales

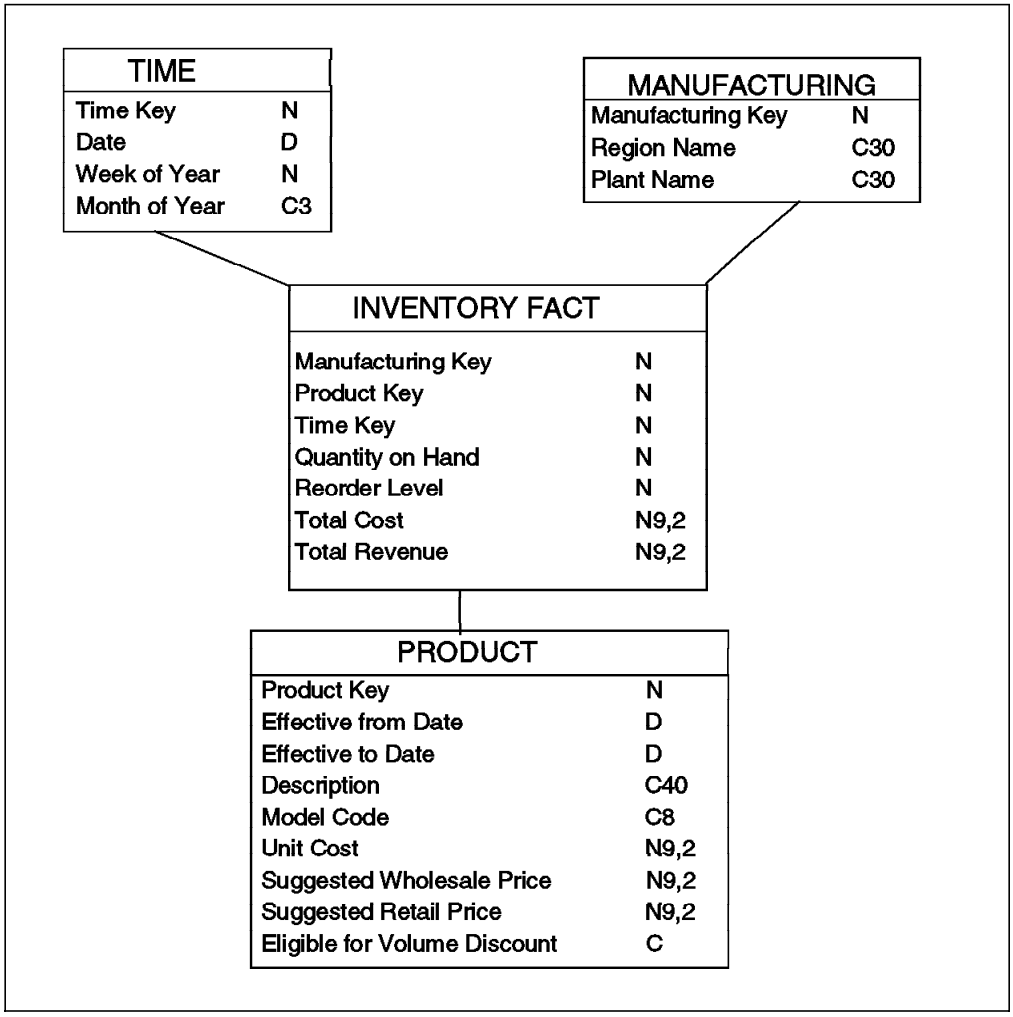


Figure 94. Dimensional Model for CelDial Product Inventory

A.6 CelDial Metadata - Proposed Solution

No model is complete without its metadata. We include here a sample of metadata that could be used for our proposed solution. It is not complete, but it provides much of the needed metadata. It is left as an exercise for the reader to analyze the sample and try to determine additional metadata required for a complete solution.

- **MODEL METADATA**

Name:	Inventory
Definition:	This model contains inventory data for each product model in each manufacturing plant, on a daily basis.
Purpose:	The purpose of this model is to facilitate the analysis of inventory levels.
Contact Person:	Plant Manager
Dimensions:	Manufacturing, Product, and Time
Facts:	Inventory
Measures:	Quantity on hand, Reorder level, Total cost, and Total Revenue

Name: Sales

Definition: This model contains sales data for each product model, on each order, on a daily basis.

Purpose: The purpose of this model is to facilitate the analysis of product sales.

Contact Person: Regional Sales Manager

Dimensions: Customer, Manufacturing, Product, Seller, and Time

Facts: Sale

Measures: Total cost, Total revenue, Total quantity sold, and Discount amount

- **FACT METADATA**

Name: Inventory

Definition: This fact contains inventory data for each product model in each manufacturing plant, on a daily basis.

Alias: None

Load Frequency: Daily

Load Statistics:

- Last Load Date: N/A
- Number of Rows Loaded: N/A

Usage Statistics:

- Average Number of Queries/Day: N/A
- Average Rows Returned/Query: N/A
- Average Query Runtime: N/A
- Maximum Number of Queries/Day: N/A
- Maximum Rows Returned/Query: N/A
- Maximum Query Runtime: N/A

Archive Rules: Data will be archived after 36 months, on a monthly basis.

Archive Statistics:

- Last Archive Date: N/A
- Date Archived to: N/A

Purge Rules: Data will be purged after 48 months, on a monthly basis.

Purge Statistics:

- Last Purge Date: N/A
- Date Purged to: N/A

Data Quality: Inventory levels may fluctuate throughout the day as more stock is received into inventory from production and stock is shipped out to retail stores and customers. The measures for this fact are collected once per day and thus reflect the state of inventory at that point in time, which is the end of the working day for a plant.

Data Accuracy: The measures of this fact are 97.5% accurate at the point in time they represent. This is based on the results of physical inventories matched to recorded inventory levels. No inference can be made from these measures as to values at points in time not recorded.

Grain of Time: The measures of this fact represent inventory levels on a daily basis.

Key: The key to an inventory fact is the combination of the keys of its dimensions: Manufacturing, Product, and Time.

Key Generation Method: The time portion of the key is simply the date the inventory level is being recorded for. The product key is retrieved from the product translation table using the product ID, model ID, and the inventory level date. The manufacturing key is retrieved from the manufacturing translation table using the region ID and plant ID.

Source:

- Name: Inventory Table
- Conversion Rules: Each row in each inventory table is copied into the inventory fact on a daily basis.
- Selection Logic: All rows are selected from the inventory table in each plant.

- Name: Sale Fact
- Conversion Rules: The rows representing sales of an individual model produce from an individual plant are summarized for each day and the result appended to the relevant inventory fact.
- Selection Logic: All rows for the day the inventory fact is being loaded with are selected from the sale fact.

Measures: Quantity on hand, Reorder level, Total cost, Total revenue

Dimensions: Manufacturing, Product, and Time

Subsidiary Facts: None

Contact Person: Plant Manager

Name: Sale

Definition: This fact contains sale data for each order which has been recorded in the sales systems at each retail store and corporate sales office.

Alias: None

Load Frequency: Daily

Load Statistics:

- Last Load Date: N/A
- Number of Rows Loaded: N/A

Usage Statistics:

- Average Number of Queries/Day: N/A
- Average Rows Returned/Query: N/A
- Average Query Runtime: N/A
- Maximum Number of Queries/Day: N/A
- Maximum Rows Returned/Query: N/A
- Maximum Query Runtime: N/A

Archive Rules: Data will be archived after 36 months, on a monthly basis.

Archive Statistics:

- Last Archive Date: N/A
- Date Archived to: N/A

Purge Rules: Data will be purged after 48 months, on a monthly basis.

Purge Statistics:

- Last Purge Date: N/A
- Date Purged to: N/A

Data Quality: It is possible for errors to be made by staff completing an order. However, the numbers recorded represent what is actually contracted with the customer and must be honored.

Data Accuracy: The measures of this fact are 100% accurate in that they represent what was actually sold.

Grain of Time: The measures of this fact represent sales of a given product on a given order.

Key: The key to a sale fact is the combination of the keys of its dimensions: Customer, Manufacturing, Order, Product, Seller, and Time. Note that Order does not have any additional attributes and thus does not physically exist as a dimension. Its primary function is to logically group product sales for analysis.

Key Generation Method: The time portion of the key is simply the date the sale takes place. The product key is retrieved from the product translation table using the product ID, model ID, and the sale date. The manufacturing key is retrieved from the manufacturing translation table using the region ID and plant ID. The seller key is retrieved from the seller translation table using the region ID, outlet ID, and salesperson ID. The customer key is retrieved from the customer translation table using the customer ID and customer location ID. The order ID from the original order is used as the order key for the sale fact.

Source:

- Name: Order Table
- Conversion Rules: Rows in each order table are copied into the sale fact on a daily basis.
- Selection Logic: Only rows for the current transaction date are selected.

- Name: Product Dimension
- Conversion Rules: The product dimension is used to calculate the cost for the product model on an order. As well, the negotiated unit price on the order is compared to the suggested retail or wholesale price (based on the outlet type) to determine if a discount was given. If so, the discount amount is calculated. If a discount is given for a product not eligible for discounting, a message is printed on an error report.
- Selection Logic: For each row being inserted into the sale fact, the product data is accessed.

Measures: Total cost, Total revenue, Total quantity sold, and Discount amount

Dimensions: Customer, Manufacturing, Order, Product, Seller, and Time

Subsidiary Facts: Inventory: the inventory fact contains daily aggregates for cost and revenue at the product and plant level of granularity.

Contact Person: Plant Manager

- **DIMENSION METADATA**

Name: Customer

Definition: A customer is any person or organization who purchases goods from CelDial. A customer may be associated with many business locations.

Alias: None

Hierarchy: Data can be summarized at two levels for a customer. The lowest level of summarization is the customer ship-to address. Data from each location (ship-to address) can be further rolled up to summarize for an entire customer.

Change Rules: New customer locations are inserted as new rows into the dimension. Changes to existing locations are updated in place.

Load Frequency: Daily

Load Statistics:

- Last Load Date: N/A
- Number of Rows Loaded: N/A

Usage Statistics:

- Average Number of Queries/Day: N/A
- Average Rows Returned/Query: N/A
- Average Query Runtime: N/A
- Maximum Number of Queries/Day: N/A
- Maximum Rows Returned/Query: N/A
- Maximum Query Runtime: N/A

Archive Rules: Customer data is not archived.

Archive Statistics:

- Last Archive Date: N/A
- Date Archived to: N/A

Purge Rules: Customers who have not purchased any goods from CelDial in the past 48 months will be purged on a monthly basis.

Purge Statistics:

- Last Purge Date: N/A
- Date Purged to: N/A

Data Quality: When a new customer is added a search is done to determine if we already do business with another location. In rare cases separate branches of a customer are recorded as separate customers because this check fails. Until such time as the customer notices separate locations dealing with us such occurrences remain as originally recorded.

Data Accuracy: Incorrect association of locations of a common customer occur in less than .5% of our customer data.

Key: The key to the customer dimension consists of a system generated number.

Key Generation Method: When a customer is copied from the operational system, the translation table is checked to determine if the customer already exists in the warehouse. If not, a new key is generated and the key along with the customer ID and location ID are added to the translation table. If the customer and location already exist, the key from the translation table is used to determine which customer in the warehouse to update.

Source:

- Name: Customer Table
- Conversion Rules: Rows in each customer table are copied on a daily basis. For existing customers, the name is updated. For new customers, once a location is determined, the key is generated and a row inserted. Before the update/insert takes place a check is performed for a duplicate customer name. If a

duplicate is detected, a sequence number is appended to the name. This check is repeated until the name and sequence number combination are determined to be unique. Once uniqueness has been confirmed, the update/insert takes place.
- Selection Logic: Only new or changed rows are selected.

- Name: Customer Location Table
- Conversion Rules: Rows in each customer location table are copied on a daily basis. For existing customer locations, the ship-to address is updated. For new customer locations, the key is generated and a row inserted.
- Selection Logic: Only new or changed rows are selected.

Attributes:

- Name: Customer Key
- Definition: This is an arbitrary value assigned to guarantee uniqueness for each customer and location.
- Alias: None
- Change Rules: Once assigned, the values of this attribute never change.
- Data Type: Numeric
- Domain: 1 - 999,999,999
- Derivation Rules: A system generated key of the highest used customer key +1 is assigned when creating a new customer and location entry.
- Source: System Generated

- Name: Name
- Definition: This is the name by which a customer is known to CelDial.
- Alias: None
- Change Rules: When a customer name changes it is updated in place in this dimension.
- Data Type: Character(30)
- Domain:
- Derivation Rules: To ensure the separation of data for customers who have the same name but are not part of the same organization, a number will be appended to names where duplicates exist.
- Source: Name in Customer Table

- Name: Ship-to Address
- Definition: This is an address where CelDial ships goods to a corporate customer. It is possible for a corporate customer to have multiple ship-to locations. For retail customers no ship-to address is kept. Therefore, there can only be one entry in the customer dimension for a retail customer.
- Alias: None
- Change Rules: When a ship-to address changes it is updated in place in this dimension.
- Data Type: Character(60)

- Domain: All valid addresses within CelDial's service area.
- Derivation Rules: The ship-to address is a direct copy of the source.
- Source: Ship-to Address in Customer Location Table

Facts: Sale

Measures: Total cost, Total revenue, Total quantity sold, and Discount amount

Subsidiary Dimensions: None

Contact Person: Vice-president of Sales and Marketing

Name: Manufacturing

Definition: The manufacturing dimension represents the manufacturing plants owned and operated by CelDial. Plants are grouped into geographic regions.

Alias: None

Hierarchy: Data can be summarized at two levels for manufacturing. The lowest level of summarization is the manufacturing plant. Data from each plant can be further rolled up to summarize for an entire geographic region.

Change Rules: New plants are inserted as new rows into the dimension. Changes to existing plants are updated in place.

Load Frequency: Daily

Load Statistics:

- Last Load Date: N/A
- Number of Rows Loaded: N/A

Usage Statistics:

- Average Number of Queries/Day: N/A
- Average Rows Returned/Query: N/A
- Average Query Runtime: N/A
- Maximum Number of Queries/Day: N/A
- Maximum Rows Returned/Query: N/A
- Maximum Query Runtime: N/A

Archive Rules: Manufacturing plant data is not archived.

Archive Statistics:

- Last Archive Date: N/A
- Date Archived to: N/A

Purge Rules: Manufacturing plants that have been closed for at least 48 months will be purged on a monthly basis.

Purge Statistics:

- Last Purge Date: N/A
- Date Purged to: N/A

Data Quality: There are no opportunities for error or misinterpretation of manufacturing plant data.

Data Accuracy: Manufacturing plant data is 100% accurate.

Key: The key to the manufacturing plant dimension consists of a system generated number.

Key Generation Method: When a manufacturing plant is copied from the operational system, the translation table is checked to determine if the plant already exists in the warehouse. If not, a new key is generated and the key along with the plant ID and region ID are added to the translation table. If the plant and region already exist, the key from the

translation table is used to determine which plant in the warehouse to update.

Source:

- Name: Manufacturing Plant Table
- Conversion Rules: rows in each plant table are copied on a daily basis. For existing plants, the plant name is updated. For new plants, once a region is determined, the key is generated and a row inserted.
- Selection Logic: Only new or changed rows are selected.

- Name: Manufacturing Region Table
- Conversion Rules: Rows in each region table are copied on a daily basis. For existing regions, the region name is updated for all plants in the region. For new regions, the key is generated and a row inserted.
- Selection Logic: Only new or changed rows are selected.

Attributes:

- Name: Manufacturing Key
- Definition: This is an arbitrary value assigned to guarantee uniqueness for each plant and region.
- Alias: None
- Change Rules: Once assigned, the values of this attribute never change.
- Data Type: Numeric
- Domain: 1 - 999,999,999
- Derivation Rules: system generated key of the highest used manufacturing key + 1 is assigned when creating a new plant and region entry.
- Source: System Generated

- Name: Region Name
- Definition: This is the name CelDial uses to identify a geographic region for the purpose of grouping manufacturing plants.
- Alias: None
- Change Rules: When a region name changes it is updated in place in this dimension.
- Data Type: Character(30)
- Domain:
- Derivation Rules: The region name is a direct copy of the source
- Source: Name in Manufacturing Region Table

- Name: Plant Name
- Definition: This is the name CelDial uses to identify an individual manufacturing plant.
- Alias: None
- Change Rules: When a plant name changes it is updated in place in this dimension.
- Data Type: Character(30)
- Domain:
- Derivation Rules: The plant name is a direct copy of the source

- Source: Name in Manufacturing Plant Table

Facts: Inventory and Sale

Measures: Quantity on hand, Reorder level, Total cost, Total revenue, Total quantity sold, and Discount amount

Subsidiary Dimensions: None

Contact Person: Plant Manager

Name: Time

Definition: The time dimension represents the time frames used by CelDial for reporting purposes.

Alias: None

Hierarchy: The lowest level of summarization is a day. Data for a given day can be rolled up into either weeks or months. Weeks cannot be rolled up into months.

Change Rules: Once a year the following year's dates are inserted as new rows into the dimension. There are no updates to this dimension.

Load Frequency: Annually

Load Statistics:

- Last Load Date: N/A
- Number of Rows Loaded: N/A

Usage Statistics:

- Average Number of Queries/Day: N/A
- Average Rows Returned/Query: N/A
- Average Query Runtime: N/A
- Maximum Number of Queries/Day: N/A
- Maximum Rows Returned/Query: N/A
- Maximum Query Runtime: N/A

Archive Rules: Time data is not archived.

Archive Statistics:

- Last Archive Date: N/A
- Date Archived to: N/A

Purge Rules: Time data more than 4 years old will be purged on a yearly basis.

Purge Statistics:

- Last Purge Date: N/A
- Date Purged to: N/A

Data Quality: There are no opportunities for error or misinterpretation of time data.

Data Accuracy: Time data is 100% accurate.

Key: The key to the time dimension is a date in YYYYMMDD (year-month-day) format.

Key Generation Method: The date in a row is used as the key.

Source:

- Name: Calendar spreadsheet maintained by database administrator.
- Conversion Rules: Rows in the calendar spreadsheet represent one calendar year. All the rows in the spreadsheet are loaded into the dimension annually.
- Selection Logic: All rows are selected.

Attributes:

- Name: Time Key
- Definition: This is the date in YYYYMMDD format.

- Alias: None
- Change Rules: Once assigned, the values of this attribute never change.
- Data Type: Numeric
- Domain: valid dates
- Derivation Rules: This date is a direct copy from the source.
- Source: Numeric Date in Calendar spreadsheet

- Name: Date
- Definition: This is the descriptive date equivalent to the numeric date used as the key to this dimension. It is the date used on reports and to limit what data appears on a report. It is in the format MMM DD, YYYY.
- Alias: None
- Change Rules: Once assigned, the values of this attribute never change.
- Data Type: Character(12)
- Domain: valid dates in descriptive format
- Derivation Rules: This date is a direct copy from the source.
- Source: Descriptive Date in Calendar spreadsheet

- Name: Week of Year
- Definition: Each day of the year is assigned to a week for reporting purposes. Because years don't divide evenly into weeks it is possible for a given day near the beginning or end of a calendar year to fall into a different year for weekly reporting purposes. The format is WW-YYYY.
- Alias: None
- Change Rules: Once assigned, the values of this attribute never change.
- Data Type: Character(7)
- Domain: WW is 1-52. YYYY is any valid year.
- Derivation Rules: This date is a direct copy from the source.
- Source: Week of Year in Calendar spreadsheet

Facts: Inventory and Sale
Measures: Quantity on hand, Reorder level, Total cost, Total revenue, Total quantity sold, and Discount amount
Subsidiary Dimensions: None
Contact Person: Data Warehouse Administrator

- **MEASURE METADATA**

Name: Total Cost
Definition: This is the cost of all components used to create product models that have been sold.
Alias: None
Data Type: Numeric (9,2)
Domain: $0.01 - $9,999,999.99.
Derivation Rules: The total cost is the product of the unit cost of a product model and quantity of the product model sold.

Usage Statistics:

- Average Number of Queries/Day: N/A
- Maximum Number of Queries/Day: N/A

Data Quality: This figure only represents the cost of components. No attempt is made to record labor or overhead costs. As well, cost is calculated using the current cost at the time a product model is sold. No attempt is made to determine when the model was produced and the cost at that time.

Data Accuracy: We estimate that the cost reported for a product model is accurate to within +/- .5%.

Facts: Inventory and Sale

Dimensions: Customer, Manufacturing, Product, Seller, and Time

---

Name: Total Revenue

Definition: This is the amount billed to customers for product models that have been sold.

Alias: None

Data Type: Numeric (9,2)

Domain: $0.01 - $9,999,999.99.

Derivation Rules: The total revenue is the product of the negotiated selling price of a product model and quantity of the product model sold.

Usage Statistics:

- Average Number of Queries/Day: N/A
- Maximum Number of Queries/Day: N/A

Data Quality: This figure only represents the amount billed for product models sold. Defaults on accounts receivable are not considered.

Data Accuracy: Defaults on accounts receivable are insignificant for the purpose of analyzing product sales trends and patterns.

Facts: Inventory and Sale

Dimensions: Customer, Manufacturing, Product, Seller, and Time

---

Name: Total Quantity Sold

Definition: This is the number of units of a product model that have been sold.

Alias: None

Data Type: Numeric (7,0)

Domain: 1 - 9,999,999.

Derivation Rules: This is taken directly from the quantity sold on an order line.

Usage Statistics:

- Average Number of Queries/Day: N/A
- Maximum Number of Queries/Day: N/A

Data Quality: This figure only represents the quantity billed for product models sold. Defaults on accounts receivable are not considered.

Data Accuracy: Defaults on accounts receivable are insignificant for the purpose of analyzing product movement trends and patterns.

Facts: Sale

Dimensions: Customer, Manufacturing, Product, Seller, and Time

---

Name: Discount Amount

Definition: This is the difference between the list price for a product model and the actual amount billed to the customer.

Alias: None

Data Type: Numeric (9,2)

Domain: $0.01 - $9,999,999.99.

Derivation Rules: The discount amount is the product of the quantity of the product model sold and the difference between the suggested wholesale or retail price of the product model and the negotiated selling price. The suggested wholesale price is used if the model is sold through a corporate sales office. The suggested retail price is used if the model is sold through a retail store.

Usage Statistics:

- Average Number of Queries/Day: N/A
- Maximum Number of Queries/Day: N/A

Data Quality: A study of the discount amounts recorded has concluded that the data is being recorded correctly. However, it is possible that discounts are being offered at inappropriate times.

Data Accuracy: Discount amounts are 100% accurate with respect to actual discounts given.

Facts: Sale

Dimensions: Customer, Manufacturing, Product, Seller, and Time

---

Name: Quantity On Hand

Definition: This is the number of complete units of a product model available for distribution from a manufacturing plant at a specific point in time (the end of a business day).

Alias: None

Data Type: Numeric (7,0)

Domain: 1 - 9,999,999.

Derivation Rules: The quantity on hand for each product model for each manufacturing plant is recorded directly from the operational inventory records at the end of each business day.

Usage Statistics:

- Average Number of Queries/Day: N/A
- Maximum Number of Queries/Day: N/A

Data Quality: The quantity of a product produced and/or shipped on a given business day varies greatly. Therefore, no conclusions can be drawn about inventory levels at points in time other than those actually recorded.

Data Accuracy: The quantity on hand is 100% accurate as of the point in time recorded and only at that point in time.

Facts: Inventory

Dimensions: Manufacturing, Product, and Time

Name: Reorder Level

Definition: The reorder level is used to determine when more of a product model should be produced. More of a model will be produced when the quantity on hand for a model falls to or below the reorder level.

Alias: None

Data Type: Numeric (7,0)
Domain: 1 - 9,999,999.
Derivation Rules: The reorder level for each product model for each manufacturing plant is recorded directly from the operational inventory records at the end of each business day.
Usage Statistics:

- Average Number of Queries/Day: N/A
- Maximum Number of Queries/Day: N/A

Data Quality: Users in the manufacturing plants report that reorder levels are reviewed infrequently. Because of this, workers responsible for initiating new production of product model will often disregard relevant warnings and plan production by ″gut feel″.
Data Accuracy: The reorder level is 100% accurate as of the point in time recorded and only at that point in time.
Facts: Inventory
Dimensions: Manufacturing, Product, and Time

- **SOURCE METADATA**

Name: Order Table
Extract Method: The table is searched for orders recorded on the current transaction date. These orders are extracted.
Extract Schedule: The extract is run daily after the close of the business day.
Extract Statistics:

- Last Extract Date: N/A
- Number of Rows Extracted: N/A

- **EXTRACT METADATA**

Name: Product and Component Extract
Extract Schedule: The extract is run daily after the close of the business day and prior to the Order and Inventory Extract.
Extract Method: The transaction log is searched for changes to the Product, Product Model, Product Component, and Component tables. These changes are extracted.
Extract Steps: See 7.5.4.3, “Getting from Source to Target” on page 74.
Extract Statistics:

- Last Extract Date: N/A
- Number of Rows Extracted: N/A

Appendix B. Special Notices

This publication is intended to guide data architects, database administrators, and developers in the design of data models for data warehouses and data marts. The information in this publication is not intended as the specifications of any programming interfaces that are provided by any IBM products. See the PUBLICATIONS section of the IBM Programming Announcement for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AS/400	BookManager
DB2	IBM®
IMS	Information Warehouse
PROFS	RS/6000
System/390	VisualAge

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix C. Related Publications

The publications listed in this section are considered particularly suitable for more discussion on topics covered in this redbook.

C.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 189.

- *Information Warehouse in the Retail Industry*, GG24-4342
- *Information Warehouse in the Finance Industry*, GG24-4340
- *Information Warehouse in the Insurance Industry*, GG24-4341
- *Data Warehouse Solutions on the AS/400*, SG24-4872
- *Data Where You Need It, The DPROPR Way*, GG24-4492

C.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

C.3 Other Publications

The following publications are also relevant as information sources:

C.3.1 Books

Adriaans, P., and D. Zaantinge. *Data Mining*. Addison-Wesley, 1996.

Aiken, P. *Data Reverse Engineering: Slaying the Legacy Dragon*. McGraw-Hill, 1995.

Barquin, R., and H. Edelstein (eds.). *Building, Using, and Managing the Warehouse*. Prentice Hall, 1997.

Bischoff, J., and T. Alexander (eds.). *Data Warehouse: Practical Advice from the Experts*. Prentice Hall, 1997.

Brackett, M. H. *Data Sharing*. John Wiley & Sons, 1994.

Brodie, M. L., and M. Stonebreaker. *Migrating Legacy Systems: Gateways, Interfaces, and the Incremental Approach*. Morgan Kaufmann, 1995.

Corey, M., and M. Abbey. *Oracle Data Warehousing*. McGraw-Hill, 1996.

Devlin, B. *Data Warehousing: From Architecture to Implementation*. Addison-Wesley, 1996.

Gill, H. S., and P. C. Rao. *The Official Client/Server Computing Guide to Data Warehousing*. QUE Corp., 1996.

Hammergren, T. C. *Data Warehousing on the Internet*. ITC Press, 1997.

Inmon, W. H. *Building the Data Warehouse*. Wiley-Qed, 1990.

_____. *Information Systems Architecture*. Prentice Hall, 1992.

_____. *Using DB2 to Build Decision Support Systems*. Wiley-Qed, 1990.

Inmon, W. H., and R. D. Hackathorn. *Using the Data Warehouse*. Wiley-Qed, 1994.

Inmon, W. H., Imhoff, C., and G. Battas. *Building the Operational Data Store*. John Wiley & Sons, 1996.

Inmon, W. H., Welch, J. D., and K. Glassey. *Managing the Data Warehouse*. John Wiley & Sons, 1996.

Kelly, B. W. *AS/400 Data Warehousing: The Complete Implementation Guide*. CBM Books, 1996.

Kelly, S. *Data Warehousing: The Route to Mass Customization*. John Wiley & Sons, 1995.

Kimball, R. *The Data Warehouse Toolkit: Practical Techniques for Building Data Warehouses*. John Wiley & Sons, 1996.

Mattson, R. *Data Warehousing: Strategies, Technologies and Techniques*. McGraw-Hill, 1996.

O′Neil, P. *Database: Principles, Programming, Performance*. Morgan Kaufmann, 1994.

Parsaye, K., and M. Chignell. *Intelligent Database Tools and Applications*. John Wiley & Sons, 1993.

Poe, V. *Building a Data Warehouse for Decision Support*. Prentice Hall, 1996.

Redman, T. C. *Data Quality for the Information Age*. Artech House, 1996.

Sprague, R. H. *Decision Support for Management*. Prentice Hall, 1996.

Thomsen, E., and G. Spofford. *OLAP Solutions: Building Multidimensional Information Systems*. John Wiley & Sons, 1997.

Data Warehousing and Decision Support: The State of the Art. Spiral Books, 1995.

C.3.2 Journal Articles, Technical Reports, and Miscellaneous Sources

Agrawal, R., et al., “Modeling Multidimensional Databases,” *IBM Research Report*, Almaden Research Center.

Appleton, E. L., “Use Your Data Warehouse to Compete,” *Datamation*, May 1996.

Codd, E. F., Codd, S. B., and C. T. Salley, “Providing OLAP to User-Analysts,” E. F. Codd Associates, 1993.

Darling, C. B., “How to Integrate Your Data Warehouse,” *Datamation*, May 1996.

Erickson, C. G., “Multidimensionalism and the Data Warehouse,” *The Data Warehouse Conference*, February 1995.

Foley, J., and B. DePompa, “Data Marts: Low Cost, High Appeal,” *Information Week*, March 1996.

Gordon, K. I., “Data Warehouse Implementation,” 1996.

_____, “Data Warehouse Implementation Plan,” 1996.

_____, “The Why of Data Standards - Do You Really Know Your Data?,” 1996.

Graham, S., Coburn, D., and Carsten Olesen, “The Foundations of Wisdom: A Study of the Financial Impact of Data Warehousing,” IDC Special Edition White Paper, 1996.

Inmon, W. H., “Creating the Data Warehouse Data Model from the Corporate Data Model,” *PRISM Tech Topics*, Vol. 1, No. 2.

_____, “Data Relationships in the Data Warehouse,” *PRISM Tech Topics*, Vol. 1, No. 5.

_____, “Information Management: Charting the Course,” *Data Management Review*, May 1996.

_____, “Loading Data into the Warehouse,” *PRISM Tech Topics*, Vol. 1, No. 11.

_____, “Meta Data in the Data Warehouse,” *PRISM Tech Topics*, Vol. 1, No. 6.

_____, “Snapshots in the Data Warehouse,” *PRISM Tech Topics*, Vol. 1, No. 4.

_____, “Time-variant Data Structures,” *PRISM Tech Topics*, Vol. 1, No. 9.

_____, “What Is a Data Warehouse?,” *PRISM Tech Topics*, Vol. 1, No. 1.

Kimball, R., “Data Warehouse Role Models,” *DBMS Online*, August 1997.

_____, “A Dimensional Modeling Manifesto,” *DBMS Online*, August 1997.

Lambert, B., “Data Modeling for Data Warehouse Development,” *Data Management Review*, February 1996.

Raden, N., “Maximizing Your Data Warehouse,” parts 1 and 2, *Information Week*, March 1996.

Snoddgrass, R., “Temporal Databases: Status and Research Directions,” *SIGMOD Record*, Vol. 19, No. 4, December 1990.

Teale, P., “Data Warehouse Environment: End-to-End Blueprint,” presentation material, IBM UK Ltd. 1996.

_____, “Data Warehouse Environment: System Architecture,” presentation material, IBM UK Ltd., 1996.

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

  For a list of product area specialists in the ITSO: type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

Additive measure. Measure of a fact that can be added across all dimensions.

Associative entity. An entity created to resolve a many-to-many relationship into two one-to-many relationships.

Attribute. A characteristic of an entity.

Business subject area. Represents a particular function or area within an enterprise whose processes and activities can be supported by a defined set of data elements.

Candidate key. When an entity has more than one possible key, each key is referred to as a candidate key.

Cube. Another term for a fact table. It can represent "n" dimensions, rather than just three (as implied by the name).

Data mart. A subject-oriented, integrated, time-variant collection of data to enable decision-making for a specific group of users.

Data mining. The use of a computer application to search for and discover new insights about the business and significant relationships among the data in the data warehouse.

Data model. A representation of data, its definition, characteristics, and relationships.

Data warehouse. A subject-oriented, integrated, time-variant collection of data to enable decision-making across a disparate group of users.

Data warehousing. The design and implementation of processes, tools, and facilities to manage and deliver complete, timely, accurate, and understandable information for decision-making.

Data warehouse data model. A data model that is structured to represent data in a data warehouse or a data mart. Some characteristics that distinguish this from other data models are source to target mapping, extract rules, extract schedules, and transformation rules.

Entity. A person, place, thing, or event of interest to the organization. Each entity in a data model is unique.

Fact. A collection of related attributes, consisting of measures and context data. It typically represents a business item, transaction, or event that can be used in analyzing the business or business processes.

Fact table. A collection of facts defined by the same dimensions and measures.

Foreign key. An attribute or set of attributes that refers to the primary key of another entity.

Grain. The fundamental atomic level of data to be represented in the fact table. Typical grains that could be used, when considering time, would be day, week, month, year, and so forth.

Key. An attribute or set of attributes that uniquely identifies an entity.

Measure. A numeric attribute of a fact representing the performance or behavior of its dimensions.

Nonadditive measure. Measure of a fact that cannot be added across any of its dimensions. A typical example of a nonadditive measure is a percentage.

Primary key. The candidate key most commonly used to identify an entity.

Relationship. The business rule that associates entities.

Semiadditive measure. Measure of a fact that can be added across only some of its dimensions. A typical example of a semiadditive measure is a balance.

Index

E

F

G

H

I

K

ITSO Redbook Evaluation

Data Modeling Techniques for Data Warehousing
SG24-2238-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

SG24-2238-00
Printed in the U.S.A.