DISSERTATION

Modeling Temporal Information in Multidimensional Data Warehouses

A business related approach to implementing the time dimension optimized for data warehousing

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

o. Univ. Prof. Dipl.-Ing. Dr. A Min Tjoa
Institut für Softwaretechnik und Interaktive Systeme (E188)
Technische Universität Wien

und

Univ. Prof. Dr. Roland Wagner
Institut für Anwendungsorientierte Wissensverarbeitung
Universität Linz

eingereicht an der Technischen Universität Wien
Fakultät für Informatik

von

Dipl.-Ing. Ahmed Hezzah

MatrNr.: 9425396
Wagramer Strasse 4
1220 Wien

Wien, September 2004

Unterschrift

Kurzfassung

Ein Data Warehouse stellt eine konsistente Sicht auf die Geschäftsdaten über die Zeit zur Verfügung. Um das zu erzielen werden Daten in logischen Dimensionen dargestellt, wobei Zeit eine der wichtigsten Dimensionen ist. Die Darstellung der Zeit ist jedoch aufgrund der komplexen Natur von zeitabhängigen Problemen und der Vielfalt der starken Abhängigkeit der Zeitdimension nicht immer selbstverständlich.

Obwohl Dimensionen im multidimensionalen, von OLAP unterstützten Modell statische Informationen darstellen, müssen sie manchmal modifiziert oder neue Einträge hinzugefügt werden. Jedoch sind Updates der Zeitdimension anders als Updates anderer Slowly Changing Dimensionen und müssen deshalb auch anders behandelt werden.

Diese Dissertation stellt ein Framework für die Modellierung der Zeitdimension in Data Warehouses für unternehmensweite Informationssysteme zur Verfügung. Sie befasst sich mit den spezifischen Problemen, die beim Design der Zeitdimension für multidimensionale Data Warehouses begegnet werden, und präsentiert Design- und Modellierungsmethoden zur Darstellung von Zeit im Data Warehouse, unter Verwendung einer oder mehreren Zeitdimensionen sowie auch Datenbank-Zeitstempel. Sie behandelt auch generische Probleme, die mit dem Design und der Implementierung der Zeitdimension verbunden sind, und die für globale Geschäftsprozesse berücksichtigt werden müssen, z.B. die Darstellung von Feiertagen, Jahreszeiten, Geschäftsperioden, Erhöhung der Granularität der Geschäftskennzahlen, Berücksichtigung der Sommer-/Winterzeit und verschiedene Zeitzonen.

Die Dissertation befasst sich außerdem mit Updates der Zeitdimension und zeigt, wie herkömmliche Methoden zur Behandlung dimensionaler Updates auf die Zeitdimension angewendet oder nicht angewendet werden können. Sie zeigt auch Beispiele für übliche Struktur- und Instanzupdates, und stellt einen Algorithmus zu deren Durchführung unterstützt vom SQL Code zu deren Implementierung vor. Sie löst allgemeine Probleme der dimensionalen Updates und zeigt ihren Einfluss auf die Zeitdimension. Diese Probleme scheinen eine breite Anwendung zu haben, und doch müssen eingehendere Untersuchungen auf diesem Gebiet für realistische, zeitbasierte Analysen in unternehmensweiten Data Warehouses geleitet werden.

Außerdem untersucht die Dissertation die Steigerung der Business Performance und die Unterstützung des globalen Austausches von zeitabhängigen Informationen, indem diese zeitabhängige Data-Warehouse-Techniken auf Komponente des SAP Business Information Warehouse (BW) abgebildet werden.

Schließlich demonstriert die Dissertation die Rolle der zeitlichen Aspekte im Data Warehouse Prozeß- und Workflow-Management, und die Auswirkung von Data Warehousing auf die Unterstützung von Business Intelligence, um den Erwartungen des Unternehmens zu entsprechen. Hoffentlich ist diese Arbeit eine gute Hilfe für Sie.

Abstract

A data warehouse provides a consistent view of business data over time. In order to do that data is represented in logical dimensions, with time being one of the most important dimensions. Representing time, however, is not always straightforward due to the complex nature of time issues and the variety of strong dependence of the time dimension on the type of business.

In the multidimensional model supported by OLAP, although dimensions represent static information, they sometimes need to be updated or new entries need to be added. However, updates to the time dimension are different than updates to other slowly changing dimensions and therefore must be handled differently.

This thesis provides a framework for modeling the time dimension in data warehouses for enterprise-wide information systems. It addresses the specific issues encountered during the design of the time dimension for multidimensional data warehouses and introduces design and modeling techniques for representing time in the data warehouse by the use of one or multiple time dimensions or database timestamps. It also discusses generic problems linked to the design and implementation of the time dimension which have to be considered for global business processes, such as representing holidays, seasons, and fiscal periods, increasing the granularity of business facts, considering the observation of daylight saving time (DST), and handling different time zones.

The thesis also addresses the issues related to updating the time dimension showing how the common techniques for handling dimension updates can or cannot be used. It also gives examples for common structural and instance updates and presents an algorithm to perform them supported by the SQL code for its implementation. It resolves general issues related to dimension updates and addresses their effect on the time dimension. These problems seem to have wide application, and yet, more in-depth investigations need to be conducted in this field for real-world time-based analysis in enterprise-wide data warehouses.

Moreover, the thesis investigates enhancing business performance and supporting the global exchange of time-dependent information by mapping those temporal data warehouse techniques to components of SAP Business Information Warehouse (BW) as an enterprise data warehouse.

Finally, the thesis discusses the role of temporal aspects in data warehouse process and workflow management and the impact of data warehousing on supporting business intelligence to meet the expectations of the enterprise. Hopefully this thesis will be a good help for you.

For Mom and Dad, with love.

I would like to express my heartfelt gratitude to my Mom and Dad, whose love, support and patience made this thesis possible. This is for you, with love and respect.

It also gives me pleasure to thank my supervisor at the Vienna University of Technology Professor Doctor A Min Tjoa for giving me the opportunity to work under his supervision and benefiting from the experience and knowledge of several years.

Thanks to my colleagues at Siemens for their kind support and encouragement, especially Walter Kowarik, Johan Eibenberger, and Sabine Wentseis.

I am also indebted to my friends and colleagues over the years whose invaluable and sometimes unintentional contributions are all over this thesis, especially Stephan Eder, Georg Kreuch, Axel Polleres and Sadet Kaya-Stein.

And finally thanks to my brother Hazem Hezzah for the continuous encouragement and exchange of ideas, which has been of great benefit to my efforts.

Contents at a Glance

1 Introduction 1

2 The Data Warehouse 8

3 Online Analytical Processing 20

4 Representing Time in the Data Warehouse 34

5 Time Dimension Updates 51

6 Modeling Temporal Characteristics with SAP BW 63

7 Data Warehouse Process and Workflow Management 78

8 Conclusions 96

References 100

Table of Contents

1 Introduction 1

1.1 Background and Motivation 1
1.2 Related Works 4

2 The Data Warehouse 8

2.1 Overview 8
2.1.1 Operational and Informational Systems 9
2.1.2 What is a Data Warehouse 10

2.2 Types of Data and their Uses 11
2.2.1 Types of Data 11
2.2.2 Business Data 12
2.2.3 Metadata 14
2.2.4 Data Beyond the Scope of the Data Warehouse 15
2.2.5 Internal and External data 16

2.3 Design Techniques 17
2.3.1 Enterprise data Modeling 17
2.3.2 Historical Data 18
2.3.2.1 The Need for Maintaining an Historical Record of the Business 18
2.3.2.2 Historical Data in the Data Warehouse Architecture 19
2.3.2.3 Historical Data Volumes 19

3 Online Analytical Processing 20

3.1 Functional Requirements of OLAP Systems 20
3.1.1 Difference Between Operational Systems and Decision-oriented Systems 21
3.1.1.1 The Source of these Differences 21
3.1.1.2 Current Differences between OLTP and Analysis-based Decision-oriented Processing 21
3.1.2 Requirements of OLAP Systems 22

3.2 Multidimensional Features 24
3.2.1 Dimensions, Hierarchies, and Hypercubes 24
3.2.2 Data 25
3.2.3 Links 26
3.2.4 Formulas 26

3.3 Benefits of the Multidimensional Approach 27
3.4 The Codd Rules and Features 27

4 Representing Time in the Data Warehouse 34

4.1 Representing Time in Business Data 34
4.1.1 The Need for Timestamps 34
5.1.2 How Data is Stored 35
5.1.3 Temporal Data Structures 35

4.2 Temporal Issues 36
4.2.1 The Time Dimension 36
4.2.2 Holidays, Seasons, and Fiscal Periods 38
4.2.3 Granularity 42
4.2.3.1 Increase the Granularity of the Time Dimension 42
4.2.3.2 Add Timestamps to the Fact Table 43
4.2.3.3 Use Twin Timestamps 43
4.2.4 Daylight Saving Time (DST) 44
4.2.5 Time Zones 46
4.2.6 Examples of Temporal Queries 48

5 Time Dimension Updates 51

5.1 Techniques to Handle Dimension Updates 51
5.1.1 Overwriting 51
5.1.2 Creating another Dimension Record 51
5.1.3 Creating a Current Value Field 52

5.2 Processing During Initial Load 52
5.3 Time Dimension Updates 53
5.3.1 Structural Updates 53
5.3.1.1 Creating a New Hierarchy Level 54
5.3.1.2 Deleting a Hierarchy Level 56
5.3.1.3 Adding a New Attribute or Flag 57
5.3.2 Instance Updates 59
5.3.2.1 Setting an Existing Day to a Holiday 59
5.3.2.2 Changes in Fiscal Periods 60
5.3.2.3 Adding One or More Years 61
5.3.2.4 Deleting One or More Years 61
5.3.2.5 Changing the DST Switch Days 61

6 Modeling Temporal Characteristics with SAP BW 63

6.1 The SAP BW Information Model 63
6.2 Time Characteristics in SAP BW 65
6.3 Mapping Temporal Characteristics to SAP BW Components 66
6.3.1 SAP BW Dimension Structure 66
6.3.2 Time Zones 67
6.3.2.1 Time Conversion in SAP BW 68
6.3.2.2 Daylight Saving Time in SAP BW 70

6.3.3 Holidays in SAP BW 72
6.3.4 Data Archiving 75
6.3.4.1 Features of the Archiving Function 75
6.3.4.2 Time Restrictions for Archiving 76

7 Data Warehouse Process and Workflow Management 78

7.1 Data Warehouse Process Management 78
7.1.1 Traces and Complexity 79
7.1.2 Data-oriented Operational Data Warehouse Processes 80

7.2 Data Warehouse and Workflow Management 81
7.2.1 Workflow Management Systems 81
7.2.2 Data Warehouse Workflow Management 82
7.2.3 Process-driven Management Information Systems 84
7.2.3.1 Monitoring and Controlling in the Workflow Life Cycle 85
7.2.3.2 Process Monitoring 86
7.2.3.3 Process Controlling 87

7.3 Example of Temporal Workflow Management 88
7.3.1 Claim Handling System 89
7.3.2 Temporal Workflow Management Issues 90
7.3.3 Implementation of Temporal Workflow Management 92

7.4 The Impact of Data Warehousing on Business Intelligence 93

8 Conclusions 96

8.1 List of Conclusions 96
8.2 Summary of Contributions 97
8.3 Future Research 99

References 100

List of Figures

Figure 1.1: Sales facts and dimensions in an OLAP model 2

Figure 2.1: The operational/informational split 9
Figure 2.2: The data warehouse 11
Figure 2.3: Types of data and the scope of the warehouse 12
Figure 2.4: Relationships between internal and external data 17

Figure 4.1: Cardinalities in the ER model 34
Figure 4.2: Deposits table using timestamps 36
Figure 4.3: Employment table using timestamps 36
Figure 4.3: A multidimensional model 37
Figure 4.5: The time dimension 38
Figure 4.6: A time dimension on hourly basis 43
Figure 4.7: A fact table using timestamps 43
Figure 4.8: A fact table using twin timestamps 44
Figure 4.9: Hours on DST switch days 44
Figure 4.10: March, 30th 2003 modeled in the time dimension 45
Figure 4.11: October, 26th 2003 modeled in the time dimension 45
Figure 4.12: A time dimension considering DST 46
Figure 4.13: A model using two time dimensions for local and universal time 47
Figure 4.14: Splitting the time dimension into a date dimension and a time-of-day dimension 48

Figure 5.1: The time dimension 53
Figure 5.2: A multidimensional model 54
Figure 5.3: A time dimension on hourly basis 54
Figure 5.4: Splitting the time dimension into a date dimension and a time-of-day dimension 56

Figure 6.1: SAP BW architecture 64
Figure 6.2: Hierarchy of SAP BW time characteristics 66
Figure 6.3: Time zones in SAP BW 68
Figure 6.4: Time conversion function of SAP BW 69
Figure 6.5: Structure for DST rules 71
Figure 6.6: DST rules 71
Figure 6.7: Variable DST rules 72
Figure 6.8: Public holidays 73
Figure 6.9: Public holiday calendar 74
Figure 6.10: Factory calendar 74

Figure 7.1: The data warehouse refreshment process 79
Figure 7.2: WDW schema 83
Figure 7.3: Process state and business state 87
Figure 7.4: Temporal workflow engine functions 92
Figure 7.5: Business led data quality for data warehousing 94

List of Tables

Table 3.1: A comparison of operational and analysis-based decision-oriented information processing activities 22

Table 6.1: Time characteristics in SAP BW 65

CHAPTER 1

Introduction

1.1 Background and Motivation

As Business tends to change over time, the business data must be able to represent that change [Devlin 1997]. However, today most data modeling and application design approaches focus only on a static view of the world.

But as events can take place to change the relationship between entities, those events somehow need to be considered and represented in the data model of an enterprise information system as well. But since the majority of today's modeling tools and databases still focus on the representations of "snapshots" of the current business information and thus provide little or no support for time dependence, the designers try to add time dependence of data to the application design.

This might be adequate in operational applications, which manage only real-time data and take a view mainly of the current state of the business. A data warehouse, however, must explicitly consider the temporal aspects of the data it contains, because it must, by definition, provide an historical view of the business. Bill Inmon spawned an information revolution with what is now known as a Data Warehouse [Inmon 1996]. While operational systems are among other things designed to meet well-specified (short) response time requirements, the focus of data warehouses is the strategic analysis of data integrated from heterogeneous systems.

This business requirement has consequences for the data model design and for the architecture of the data warehouse. The absence of support for this temporal issue led data warehouse designers to take several approaches to reflect history in the database design [Snodgrass 2000].

As we move toward the multidimensional approach data is represented in logical dimensions in order to provide a consistent view of the data over time, a view that can be used by decision support systems. One of the major dimensions in every multidimensional data warehouse is the time dimension. The time dimension contains descriptive temporal information, and its attributes are used as the source of most of the temporal constraints in data warehouse queries [Kimball 1996] and [Kimball 1998]. It is obvious that the design of the time dimension is not always straightforward as it strongly depends on the type of business and the requirements of the enterprise. Other dimensions typically found in business data warehouses might be product, geography, customer and organizational structure.

Dimensional data warehousing became a research topic in the early 90's. The amount of data to store was increasing, and requirements to analyze it were getting higher, so new techniques were apparently needed in order to solve problems such as slow query performance and temporal inconsistency that existed in OLTP (Online Transaction Processing) systems. Dimensional data warehousing and OLAP (Online Analytical Processing) differ from OLTP

systems in that they are mainly focused on how to get data out from a database instead of how to put it in [Kimball 1996].

Usually, OLAP models are based on the idea of a "star schema", in which data is stored in sets of dimension and fact tables. Data in the fact tables reflect the dynamic aspect of the data warehouse (e.g. daily sales), while data in the dimension tables represent basically static information (Figure 1.1).

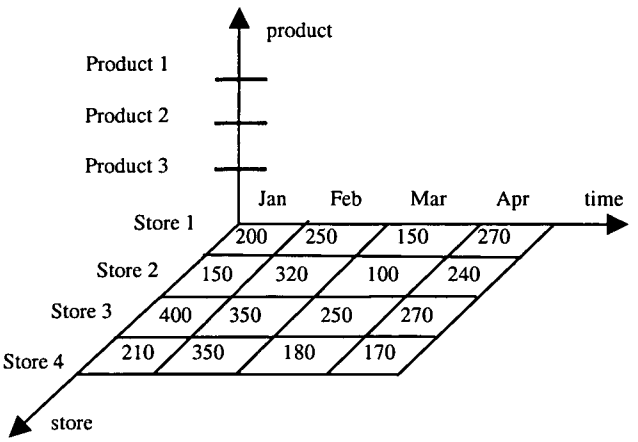


Figure 1.1: Sales facts and dimensions in an OLAP model

These dimensions, although representing static information, sometimes need to be updated or new entries need to be added. For instance, humans change their names, get married or divorced, change their addresses or phone numbers, countries unite and separate, products emerge and vanish, and organizational structures evolve. However, as these dimensions are almost constant over time and their structure can be preserved with only minor changes, they are called *slowly changing dimensions* [Kimball 1996].

Although all these dimensions are created equally in every data warehouse, the fact is that the time dimension is very special and must be handled differently from the other dimensions. Changes to the time dimension do not occur as frequently as to other slowly changing dimensions, but are rather the exception, so that we can call it *very slowly changing*. They are also different in nature as they usually do not reflect updates made in the OLTP system by some business process, as is the case with product or customer data, but are rather due to decisions made by the management, for instance changes to the fiscal periods or adding new holidays.

Another important aspect of the time dimension is that most changes require more than one dimension record to be updated; fiscal periods cover several weeks or months, and holidays and seasons are repeated every year. This is different than other slowly changing dimensions like product and customer where a product or a customer is represented by one single record.

Also after updating a time dimension record, the old value will no longer be used in the fact table unlike other dimensions such as product, where there might still be products with old attribute values (e.g. package type) still selling in the stores after the update. There is no overlapping here and also no need to consider a transition period.

The aim of this thesis is to introduce a specification of the time dimension in enterprise data warehouse systems, which is consistently applicable for handling the analysis of global enterprise data. The problems arising in multinational corporate groups when combining data with a temporal dimension are enormously cost-intensive. Even the minor problem of daylight saving time DST for one of the world-wide leading energy companies could cause data warehouse costs of millions of dollars.

Chapter 2 provides an overview of the basic concepts of data warehousing and how the informational environment differs from the operational environment. It describes the different types of data and identifies those appropriate for data warehousing. This analysis sets the scope of the warehouse and classifies the types of data according to their usage. Finally, it introduces design techniques such as enterprise data modeling and representing historical data in the warehouse.

Chapter 3 describes the basic features and concepts of OLAP and multidimensional technology. It presents an overview of multidimensional features, such as dimensions, hierarchies, hypercubes, links, and formulas, but also addresses the functional requirements of OLAP systems and the benefits of the multidimensional approach to design and work with analysis-based decision-oriented systems, referring to the Codd rules and features as described by [Codd 1993].

Chapter 4 deals with the subject matter related to representing time in the data warehouse. It discusses the design of the time dimension and introduces design techniques for its implementation. It presents a practical approach, which also models relevant real-world business issues such as holidays, seasons, and fiscal periods by extending the time dimension with new attributes and flags. It uses the time dimension together with timestamps to resolve major granularity issues. It also discusses the observation of day saving time DST and its effect on the design of the time dimension by providing an approach to model days on which time is shifted to DST and backwards. Finally, it addresses issues related to having different time zones and demonstrates how the use of local and universal time can resolve these issues.

Chapter 5 addresses the issues related to updating the time dimension as a slowly changing dimension. It presents the differences between time and other slowly changing dimensions such as product, customer or store. First it introduces the common techniques used to handle dimension updates as described in [Kimball 1996]. Then it demonstrates how these techniques can or cannot be used on the time dimension, giving examples for common structural and instance updates. It introduces an algorithm to perform these updates supported by the SQL code for its implementation. Finally, it resolves general issues related to dimension updates and addresses their effect on the time dimension.

Since SAP Business Information Warehouse (BW) is today a suitable and viable option for enterprise data warehousing, Chapter 6 introduces a mapping of the data warehouse techniques discussed in the previous chapters to components of SAP BW like InfoCubes and master data tables.

Chapter 7 discusses the role of temporal aspects in the data warehouse process and workflow management with focus of the development of workflow data warehouse and the implementation of temporal workflow management. It presents an example of a claim handling system and addresses the impact of data warehousing on supporting business intelligence to meet the expectations of the enterprise.

Hopefully this thesis will help you to solve these and many other issues that you will encounter as you design, implement and maintain temporal data warehouse applications.

1.2 Related Works

One of the first approaches which incorporates the notion of time for modeling enterprise data is described in [Eder 1987], where timestamps and states of entities and relationships are introduced. This approach leads to the specification of business rules with situation/activation diagrams as described in [Lang 1997]. A detailed description of most current research performed for this purpose can be found in [Wijsen 1999] and [Wijsen 2003].

The functions needed to implement a data warehouse architecture including different types of data are described in [Devlin 1997] and [Inmon 1996]. It addresses the use of timestamps to store periodic and historical business data, but doesn't consider the multidimensional approach widely used today. This is more discussed in [Kimball 1996] and [Kimball 1998] with case studies of data warehouses for different types of businesses, almost all using a daily-based time dimension, unfortunately with no focus on the issues related to its implementation.

Adding history to the temporal database application is investigated in [Snodgrass 2000] with focus on issues related to valid and transaction time, intervals and periods and state tables for valid and transaction time. It also presents some implementation considerations for the temporal database logical and physical design and demonstrates application development issues using SQL.

In [Allen 1983] John Allen introduces a temporal logic based on intervals and their qualitative relationships in time. Allen's approach is simple, transparent, and easy to implement. The basic elements of Allen's theory are intervals corresponding to events (rather than points corresponding to instants), qualitative relations between these intervals, and an algebra for reasoning about relations between intervals.

A conceptual multidimensional data model, which facilitates even sophisticated constructs based on multidimensional data units or dimension members, is introduced in [Nguyen 2000]. This model is able to represent and capture natural hierarchical relationships among dimension members within a dimension as well as the relationships between dimension members and measure values and is modeled using UML. Dimension updates are formally discussed in [Vaisman 2002].

[Bruckner 2001] presents an approach for modeling conceptual time consistency problems and introduces a model that deals with timely delays. However, this model does not address issues related to time dimension updates as much as data consistency and general updating issues. Changes of dimension data are discussed in [Eder 2002a], which presents an approach to represent temporal behavior of master data within existing, non-temporal data warehouses.

In [Ravat 2000] a data warehouse class concept is introduced, which is based on the concepts of temporal filter and archive filter. In order to define the warehouse class structure, the warehouse class population, and the warehouse class hierarchy, it defines mapping functions to specify derived, calculated, and specific properties, and organize the inheritance hierarchy of the warehouse classes, allowing extracting only relevant data. In [Yang 2000] Yang and

Widom study incremental maintenance of temporal views using a temporal query language equivalent to TSQL2. Although [Ravat 2000] does not organize data multidimensionally, it provides a more flexible temporal model than [Yang 2000] because the purging values are not deleted, but they are archived.

The use of multiple time dimensions is mentioned in [Kimball 1999], which introduces the concept of a data webhouse. It uses a clickstream data mart to store all web activities for later analysis of user behavior. This is also discussed briefly in [Pedersen 2001] with focus on the influence of the web on data warehousing, but also the design of clickstream fact tables and dimension tables.

Other temporal issues like fiscal periods and granularity are briefly discussed in [Kimball 2002] and [Kimball 1997] with more focus on using the time dimension to resolve this issues, but design issues are not investigated in detail.

Although a lot of research has been done in the field of data warehousing and temporal databases, very little focus was given to the issues related to handling time dimension updates. Most researchers (e.g. [Inmon 1996], [Kimball 1996], [Vaisman 2002]), when talking about slowly changing dimensions, focus more on products, stores, regions, but not on the time dimension.

[Hurtado 1999] introduces a formal model supporting dimension updates in a multidimensional model, and presents a collection of primitive operators to perform them. [Mendelzon 1999] extends this model, adding a set of semantically meaningful operators, which allow performing some usual dimension updates in a more efficient way. It also formally defines the mapping from the multidimensional to the relational model, and studies how an implementation of the dimension updates could proceed, in a ROLAP storage of the data warehouse.

[Moody 2000] describes a method for developing dimensional models from traditional entity relationship models that can be used to design data warehouses and data marts based on enterprise data models. Understanding the design alternatives presented here helps understand how to perform updates on the different models.

[Bliujute 1998] handles problems related to the management of changing information, namely handling slowly changing dimensions and state-oriented data, by presenting a new approach to data modeling in data warehouses, so-called temporal star schemas. The main difference between the temporal star schema and the regular star schema is in their treatment of time. In the temporal star schema, time is not a dimension and it is represented as one or more attributes in all tables.

[Eder 2001] proposes an extension of the multidimensional data model employed in data warehouses allowing coping correctly with changes in dimension data. The approach provided in [Vassilidis 1998] is based on the notion of the base cube, which is used for the calculation of the results of cube operations. Its focus is on the support of series of operations on cubes, i.e. the preservation of the results of previous operations and the applicability of aggregate functions in a series of operations. It also provides a mapping of the multidimensional model to the relational model and to multidimensional arrays.

In [Hezzah 2004a] we addressed the specific issues encountered during the design of the time dimension for multidimensional data warehouses. We introduced design and modeling techniques for representing temporal information in the data warehouse and also discussed problems linked to the design and implementation of the time dimension, such as representing holidays, seasons and fiscal periods, increasing the granularity of business facts, considering the observation of daylight saving time (DST) and handling different time zones.

In [Hezzah 2004b] we went one step further and addressed the issues related to structural and instance updates to the time dimension and showed how they differ from other slowly changing dimensions, such as product, customer and store. [Eder 2002b] analyzes problems related to the change of the structure of dimensions, i.e. the content and relationships of master data like the diagnostic codes, or other key values. It shows how to superimpose conventional multidimensional data warehouses with temporal master data to allow queries spanning multiple periods to return correct answers.

[Hezzah 2004c] provides an attempt to representing temporal information using SAP Business Information Warehouse (BW) as an enterprise data warehouse. It shows how business performance can be enhanced by applying the concepts introduced in our previous works to the SAP BW temporal information model.

[Hezzah 2005] investigates in more depth the information model of SAP BW and focuses on the time characteristics found in this model. It shows how the global exchange of time-dependent business information can be supported by mapping temporal concepts introduced in our previous works to SAP BW components with focus on issues related to the dimensional representation of time, conversion of local times from different time zones, and modeling relevant real world business issues such as holidays, seasons and daylight saving time (DST). It also investigates the time restrictions on the data archiving function of SAP BW.

A detailed overview of the management of the data warehouse using SAP BW is provided in [Prosser 2001] with focus on data maintenance issues. [McDonald 2002] shows how to implement the SAP Business BW and create useful applications for business analysis of company-wide data. It focuses on the business content and options available when trying to deliver value from the data stored in the SAP BW, and it includes a methodology for implementing the BW, such as data modeling and techniques for capturing and transforming data.

[Egger 2004] delivers useful information that is integral to understanding and leveraging the full potential of SAP BW from data modeling to optimizing data collection to maximizing reporting capabilities. It provides a detailed introduction to SAP BW: InfoObjects, InfoProvider, InfoCubes, star schema, DataSources, InfoSources, Web Items, etc.

The aim of this thesis is to give a framework for modeling the time dimension in data warehouses for enterprise wide (global) information systems with focus on its applicability for practical issues, such as daylight saving time (DST) or problems related to time zones, holidays, and fiscal periods. The thesis presents design techniques for representing time in multidimensional data warehouses and temporal databases. It provides an approach to model practical problems of different time representations, which will be demonstrated considering relevant real-world examples, such as the DST-problem, the modeling of holidays, fiscal periods, etc., by using one or multiple time dimensions and timestamps.

Furthermore, the thesis addresses the issues related to updating the time dimension as a slowly changing dimension. It presents the differences between time and other slowly changing dimensions such as product, customer or store, and introduces an algorithm to perform structural and instance updates to the time dimension.

Moreover, it investigates enhancing business performance and improving the global exchange of time-dependent business information by mapping the temporal data warehouse concepts introduced throughout the thesis to SAP BW components like InfoCubes and master data tables.

Finally, the thesis discusses the role of temporal aspects in data warehouse process and workflow management and the impact of data warehousing on supporting business intelligence to meet the expectations of the enterprise.

CHAPTER 2

The Data Warehouse

2.1 Overview

The concept of a data warehouse springs from the combination of two sets of needs:

- The business requirement for a better company-wide view of information
- The need of the information system (IS) department to manage company data in a better way

Considering only the business need for a company wide-view of information can lead to solutions based on allowing any user to access any data wherever it resides. However, such solutions ignore the fundamental distinction between data and information and are thus unacceptable. Actually what business users require is information, which is often defined as data in its business context. Applications are usually built to contain data divorced from a business context. Such data is simply unsuitable for direct use by end users.

On the other hand, the IS need for improved data management, when taken alone, seems to present no more than an IS cost-reduction or technology implementation project that due to its implementation cost always comes at the bottom of the priority list.

However, combining both sets of needs gives a new perspective. If the IS needs for data management were addressed, the business need for a company-wide view of data would be far easier to meet. Similarly the need for a company-wide view of data is required for solving the data management problem. It was the user and IS needs why the concept of a data warehouse evolved. The data warehouse has been approached many times and from many directions in the last decade, but its full potential could not be realized because no comprehensive methodology existed and because software tools lacked the necessary function.

Until the mid-1970s computing was the preserve of the IS shop and the end user was a rarity due to the complexity of hardware and software at that time. Business people always looked to someone else to provide them with the information they needed for decision making.

By the mid-1980s end users with the ability to deal with both the business and technical aspects of data became more common. They were distinguished by some characteristics as they were:

- Familiar with business terms
- Driven by real business needs to solve existing problems or to find new opportunities
- Aware of the value of "real" information in decision making
- At ease using technology to meet their goals
- Open to "do-it-yourself" solutions but keen to avoid repetition
- Understand the meaning of data in current applications

Here it is necessary to know the difference between data and information. While data is the computerized representation of business information, in terms of tables, arrays, pointers, etc., information is a representation of the business understood and used by end users.

2.1.1 Operational and Informational Systems

Early attempts to support decision making led to a partitioned view of the business data: one part dedicated to running the business at a detailed level and the second part focusing on managing the business at a summary level. By the end of the middle ages, a distinction had evolved between the operational (or production) systems and the informational (or decision support) systems.

Operational systems have the following characteristics as described by [Devlin 1997]:

- They run the business on a second-to-second basis.
- The data they contain is a current and largely real-time representation of the state of the business.
- Individual events (or transactions) in these systems are generally limited in scope, are rather simple, and often result in an update of the data.
- They are optimized for fast response time for predefined transactions, and have a special focus on performance of update transactions.
- They are used by people who deal with customers or products on an individual level, for example, clerks, salespeople, and administrators.
- They are increasingly used by customers themselves.

The characteristics of informational systems are as described by [Devlin 1997]:

- They are used to manage and control the business.
- The data is historical or point-in-time; that is, it represents a stable view of the business over a period of time or at a particular point in time.
- Optimization is for inquiry rather than update.
- The use of these systems is loosely defined and may be wholly unpredictable.
- They are used by managers and end users to understand the business and make judgments and decisions based on this knowledge.

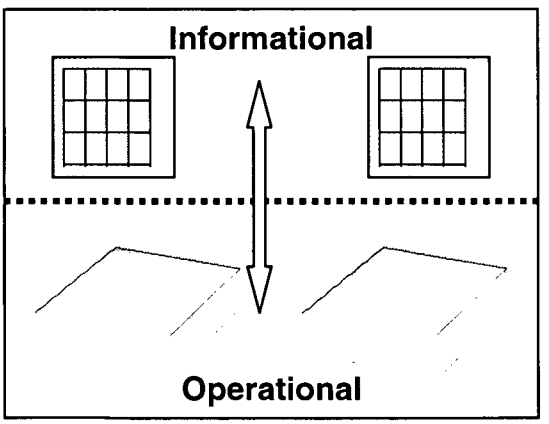


Figure 2.1: The operational/informational split

As shown in Figure 2.1 [Devlin 1997] the boundary between the two categories is not very clear, but it is distinct enough to be useful. In fact it provides a common starting point for all discussions of data warehousing. It should not be concluded from this discussion that any set of existing applications can be easily categorized into two well-defined groups. Because of the way applications have developed, informational components may have been added to previously developed operational applications. However, in many cases it is fairly easy to identify which parts are operational and which are informational.

2.1.2 What Is a Data Warehouse

In [Devlin 1997] Barry Devlin gives a definition of a data warehouse as "a single, complete, and consistent store of data obtained from a variety of sources and made available to end users in a way they can understand and use in a business context".

Bill Inmon says in [Inmon 1996]: "The goal of the data warehouse is to make accurate data, which is consistent across the enterprise, accessible to end-users in an efficient way, which is impossible when the data resides on an operational system."

In [Thomsen 1997] Erik Thomsen says: "A data warehouse is not a piece of software you can buy. It is a process of reengineering the information flow within the organization", and therefore he suggests using the term "data warehousing" instead as he sees the value of a data warehouse in "delivering the information to the end user not in the amount of data stored in it".

Achieving completeness and consistency of data in todays IS environment is not that simple. The first step is to know what data is required. This data exists today in various sources on different platforms, and must be copied from these sources for use in the data warehouse. It must be combined according to the enterprise model, even though it was not originally designed to support such integration. After that it must be cleansed of structural and content errors. This step – known as populating the data warehouse - is recognized as one of the most important, but most difficult technical aspects of warehouse implementation.

The data thus copied and transformed according to the enterprise model is stored in the data warehouse. In order to be understood and used in a business context, this data must be transformed into information. Therefore, the end user needs a catalog which describes the data in its business context. This catalog acts as a guide to the location and use of the information.

Finally, end users require a set of tools to analyze and manipulate the information thus made available. These tools provide the interface between the user and the information and are the final step in the overall transformation of raw data into useful and usable information.

These major components of the date warehouse are shown in Figure 2.2 [Devlin 1997].

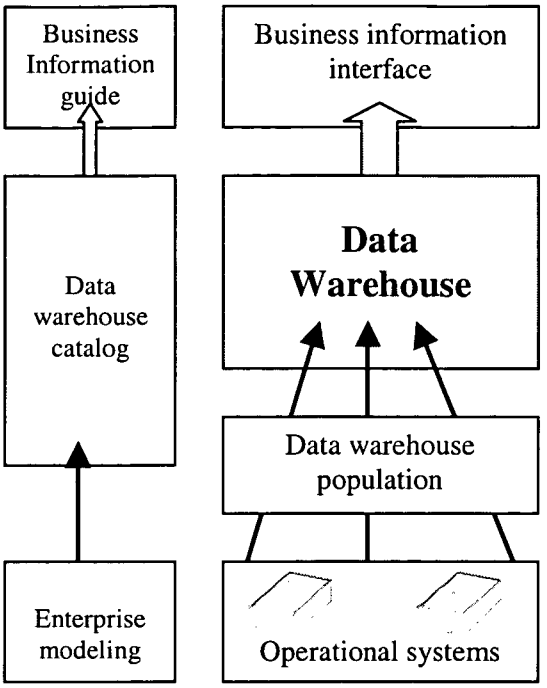


Figure 2.2: The data warehouse

2.2 Types of Data and their Uses

There are many varieties of data stored in computers today. Some types of data are particularly appropriate for storage and management in a data warehouse. Others are not. The purpose of this section is to distinguish between the different types of data and to identify those appropriate for data warehousing.

2.2.1 Types of Data

Data is defined as a computerized representation of business information. At the highest level data can be partitioned in many ways according to its:

1. Meaning:
   Data may have intrinsic meaning or may be a representation of something that has meaning. This distinction is the most fundamental and perhaps the most difficult to understand.
   Computer-based data has long been used to run and manage a business. Such data, called business data, represents the state of the business, and its value lies in the meaning it represents.
   Another type of data is growing rapidly in importance. This data has its own intrinsic meaning, and its value lies in its content rather than in what it represents. Thus it is termed data as a product, because it is produced, bought, and sold in the same way as any physical product. Examples are digitally stored movies or books.
   Finally there is metadata, which describes the meaning of data. Such metadata exists

only to define or describe business data or data as a product.

2. Structure:
   Data may be highly structured, consisting of many well-defined interrelated fields or records, or unstructured, where the internal structure is very variable, or it may fall anywhere between these two extremes.

3. Scope:
   Data may be personal, where its owner can change it as he or she pleases, or public, where its use is shared among a number of people and any changes require careful management.

Figure 2.3 [Devlin 1997] shows that data warehousing focuses on business and metadata that are mainly public in scope, and covers both structured and unstructured components of business data and metadata. We can say that data warehousing provides the self-consistent and well-understood data needed to manage the business both as a whole and in its individual parts.

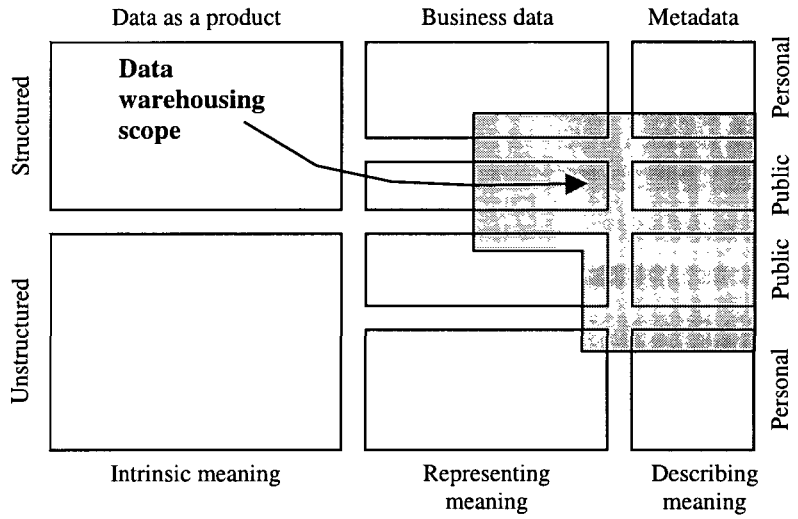


Figure 2.3: Types of data and the scope of the warehouse

2.2.2 Business Data

"Business data is the data required to run and manage an organization, typically a business organization. It represents the activities that the business undertakes and the objects in the real world, customers, locations, and products, with which it deals. It is created and uses through transaction processing systems and decision support systems" [Devlin 1997].

There are four criteria used to determine types of business data:

1. Usage in the business:
   Operational data is used to run the business and is related to short-term actions or decisions.
   Informational data is used to manage the business in the longer term.

2. Scope of the data:
   Data may represent a single item or transaction, or it may be a summary of the net effect of a set of items or transactions.
   Detailed or ***atomic data*** is critical to running the business as it often focuses on basic objects or transactions such as products, orders or customers.
   Summary data is used in managing the business and showing a broad view of the way the business is operating.

3. Read/write versus read-only data:
   Read/write data requires careful design of the update process to ensure that business integrity rules are obeyed. Its structure is optimized for writing to databases or files.
   Read-only data is usually designed with unplanned inquiry in mind and provides a stable base for repeated reading.

4. Data currency:
   Current data is a view of the business at the present time. It is up to the second and is subject to change over the time based on business activities. It presents an accurate representation of the current performance of the business.
   Point-in-time data is a stable snapshot of the business data at a particular moment in time and reflects the status of the business at that moment.
   Periodic data is an important class of data that provides a definitive record of the business as it changes over a period of time. Such periods of time are of various duration, but the time period covering a number of years is of particular interest in data warehousing.

These criteria allow us to identify three types of business data:

1. ***Real-time data*** is current or up-to-the-second data representing the current status of the business and is used to run the business. It occurs on a detailed level and is accessed in read/write mode, usually through predefined transactions. In general, real-time data is the data created, manipulated and used by operational or production applications. It is controlled and managed by the IS department and is usually unstructured as a result of old systems and repeated maintenance.

2. ***Derived data*** is point-in-time or periodic data, at a detailed or summary level, that is derived by some process from real-time data and used to manage the business. It is the set of data that is provided to the end user and used for decision support.

3. ***Reconciled data*** is a special type/category of derived data that occurs at an historical, detailed level and is designed and used to ensure consistency of data across the entire enterprise. It is thus a very important element of the data warehouse. Examples include data from multiple applications with different structures or formats that has to be combined for the use in the data warehouse.

Unstructured Business Data

Management information systems have traditionally focused on well-structured data. Such data commonly has the following characteristics:

- A significant proportion of the data is numerical.
- There are multiple attributes for each entity (expressed as multiple fields per record or multiple columns per table).
- Multiple relationships exist between different entities.
- Most of the individual attributes are small in size.

On the other hand there is unstructured data, which have opposite characteristics to those listed above. Image, audio and video are examples of highly unstructured data. Examples are images of insurance forms or a driving license. Textual data such as notes and documents fall between the two extremes.

The importance of less structured types of data is rapidly increasing in all businesses and information systems. As it falls under business data, it is used to run or manage the business on a summary or detailed level. It can be read/write or read-only, but it is less time-sensitive than structured data.

Unstructured data should be included in the data warehouse, but only after structured data is well supported.

2.2.3 Metadata

"Metadata is data that describes the meaning and structure of business data, as well as how it is created, accessed and used" [Devlin 1997]. Since business data doesn't exist but rather has to be created, maintained and accessed through business processes that are implemented through applications, the business needs a full description of its business data and the processes by which to maintain and use it. Metadata thus describes a number of aspects of the business and of the corresponding application functions.

As for business data, metadata can be classified according to some basic criteria:

1. Alignment to the application life cycle:
   Build-time metadata is designed to facilitate consistency of use, as well as reuse of both data and function by application and database designers. ***Production-time metadata*** is designed to facilitate finding, understanding and using the required data in the business.
   Build-time metadata is the primary (although not the only) source for production-time metadata, and both can be used in the operational and informational environment.

2. Active or passive use:
   This characteristic describes the technical use made of production-time metadata. Metadata that is used to control the action or function of some application has an ***active*** role.
   Metadata used in look-up mode to find some business data or to understand some characteristic of that business data is being used in ***passive*** mode.

Applying these criteria to metadata leads to defining three types of metadata:

1. ***Build-time metadata*** is the metadata created and used in the process of application and database design and construction. According to the definition of data warehousing

scope, build-time metadata is outside the scope of the warehouse. However, as for real-time business data, build-time metadata is the source of the metadata that does fall within the scope of the warehouse.
Build-time metadata is stable in comparison with the business data it describes. It changes only when the overall structure of the business or its implementation in application changes.

2. ***Control metadata*** is used to control the operation of the data warehouse and its infrastructure. Therefore it is part of the production-time metadata.
There are two types of control metadata: *currency metadata*, which describes information about the currency or timeliness of the business data in form of timestamps, e.g. the time of last update of a table in a database or the run times of applications, and *utilization metadata*, which is associated with the security and authorization functionality used to control access to the warehouse.

3. ***Usage metadata*** is metadata structured to support end users' use and understanding of business data. It is sourced from build-time metadata and is similar in content, but differently structured to meet end users' rather then application and database designers' needs. Usage metadata describes the following aspects of the data or application: meaning in the business, ownership and stewardship, data structure, and application aspects.

Structured and Unstructured Metadata

While the distinction between structured and unstructured is strong for business data, this is not so for metadata. Unstructured metadata may play a significant role throughout the implementation of the warehouse. Such metadata consists mainly of free-form textual descriptions of data and processes in the business and it exists side by side with more structured metadata such as table names or relationship definitions.

2.2.4 Data Beyond The Scope of The Data Warehouse

In the previous sections we described the types of data included in the scope of data warehousing. Now it is appropriate to outline the types of data excluded from that scope and the reasons for that. We identify two types of such data:

1. **Data as a product** is data with intrinsic meaning and value that is designed to be bought and sold, as with any physical product. For example, the value of this thesis lies in its information content. As a product, it is today produced on paper. However, for most of its production process, it existed as textual and image data in a computer. In the future it may be considered worthwhile to produce an electronic version of this thesis that can be bought and sold. Also audio and video products such as movies and music recordings are examples for data as a product.

   Such data is different in structure and content from business data, but the main difference is rather in the way such data is used in the business. "While the value of traditional business data lies in how well it reflects the reality of business activities, the value of data as a product is intrinsic to its content, and therefore it must be managed in a different way from business data" [Devlin 1997].

Data as a product is mainly unstructured, but there are also some examples for structured data as a product, such as market research companies that analyze market trends and produce and sell the data output from these analyses. This data is structured and in its content is very similar to the data that companies use to run and manage their businesses. Thus, this data is transformed into business data when the purchasing company uses it as a part of its management information.

Similarly some business data of a company might be of intrinsic value, allowing it to be sold and thus transforming business data into data as a product, such as the customer database of a telecommunications company.

2. **Personal business data and metadata** is data under the control of one person, which he or she can delete or change as needed, without having to consult anybody else on the organization. This includes personal notes, addresses, contact information, to-do lists, etc.

   As LAN and client-server technology improves the movement of data between the desktop and the corporate IS environment increases. This way sharing personal data and copying corporate data to the desktop becomes easier.

   However, since this data cannot be controlled or managed by the IS department, it cannot stand over its quality or integrity.

As a consequence, data as a product and also personal business data is outside the scope of the data warehouse.

2.2.5 Internal and External Data

In the past, the most of data of interest to an organization was internal within that organization. Source of external data were small enough and data volumes low enough that the impacts of external data on the overall architecture were relatively insignificant. This has changed significantly, especially with the development of the Internet, which has also caused an exponential growth in the volume of data entering and leaving an organization.

As shown in Figure 2.4 [Devlin 1997] external data that enters an organization can be divided into the following categories:

1. **Structured business data** can be easily combined with existing internal data and thus has to be handled with great care to ensure its quality and consistency with existing data. The associated metadata must be made available as well.

2. **Unstructured business data** is harder to embed automatically in the decision-making process and thus the dangers it causes are smaller.

3. **Data as a product** enters the data warehouse as business data (as described in the previous section) and is therefore handled the same way like external business data. Internal business data might also be transformed to data as a product before leaving the organization as previously mentioned.

4. **Metadata** usually doesn't leave the organization unless it accompanies business data. This is needed so that business data can be understood in context and reconciled as required.

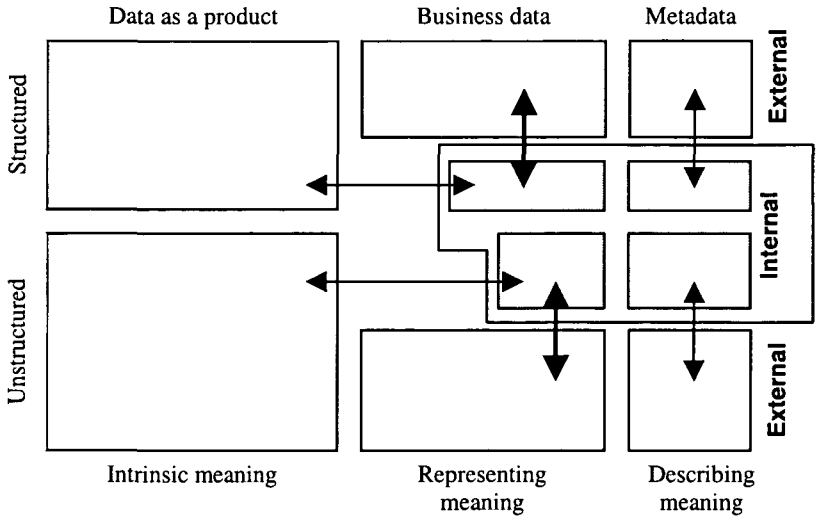


Figure 2.4: Relationships between internal and external data

This analysis is of some significance when planning to link the organization to the Internet. Free access to external data such as Internet resources must be regulated by organizational procedures to maintain internal data quality and consistency.

2.3 Design Techniques

Designing a data warehouse requires the use of different techniques than those used to design operational system or traditional informational applications. This springs from three characteristics of the data warehouse as described by [Devlin 1997]:

1. The scope of the data warehouse eventually encompasses the whole enterprise.

2. The data warehouse contains the historical record of the business.

3. The source for all data in the warehouse is existing data, which may be changing in structure and content, and of variable quality.

These characteristics lead directly to the design techniques described in this section.

2.3.1 Enterprise Data Modeling

Enterprise data modeling is a design technique that defines the contents of the warehouse and allows the entire scope of the business to be included in the data warehouse. This process cannot be achieved in a single step, but rather must be addressed in stages. This requires a combination of the business data model and the business process model. The former is of far greater significance in the informational environment.

Application-level modeling has been used in the development of specific business functions within the scope of a single application. It provides a logical view of the data required by the application, driven and derived by users' needs. In the context of application development this approach allows both the potential users of the application and its developers to focus only on the data and processes required and to do so in a structured manner.

However application-level modeling provides no significant support for integrating applications or for combining data from different sources. As this is of significant importance in the informational environment, a broader type of modeling is required, known as enterprise modeling.

The focus of **enterprise modeling** is a complete and integrated view of all the data and processes in the business. It attempts to treat the data entities at the most general level, so that all commonality in the business data is made visible and usable. While an application-level model's purpose is the design of a single application, the enterprise model has broader aims. These include:

1. providing a single systems-development base and supporting the integration of existing applications.
2. supporting the sharing of data between different areas in the business.
3. enabling effective management of data resources by providing a single set of consistent data definitions.
4. supporting the creation and maintenance of company-wide management information.

Enterprise modeling is used in the operational environment to attempt to re-architect a set of existing and diverse applications that were never designed to work with one another in the first place. These changes are usually very expensive and as a result unsuccessful.

Thus, enterprise modeling for the warehouse still provides a considerable challenge to most companies. Because the scope covers the whole enterprise, the size of the project will require careful management to ensure that it delivers results in a reasonable time period. The people needed here are those with both a broad understanding of how the business works and a detailed knowledge of a particular area of the business. They should also process a vision of how the business should work, as this is what modeling is all about.

2.3.2 Historical Data

One of the reasons for developing the data warehouse approach is the need for storing historical data; data which is no longer current, but which has been current for a certain period in time. This kind of data plays a significant role in data warehouses used for trend analysis as it defines a complete record of the business.

2.3.2.1 The Need for Maintaining an Historical Record of the Business

Requirements for maintaining an historical record of the business fall into two broad areas:

1. A view of the business at a given time
   This is provided by snapshot data, which gives the end users different views of the business at different times.

2. Business trend analysis
   Using periodic data end users can undertake trend analyses at any level of granularity: monthly, daily, etc.

2.3.2.2 Historical Data in the Data Warehouse Architecture

If we think of a three-layer architecture of the data warehouse which consists of a real-time layer, a reconciled layer and a derived layer, historical data would be a mix of all types of data and would reside in any or all of the three layers.

Real-time data represents the current state of the business, but a view of the events that lead to this up-to-minute status is still needed. The time span for this depends on the type of business.

Derived data is used to manage and analyze the business and may consist of periodic or snapshot data. The type and volume of historical data required depends on the business needs of the users.

Reconciled data support enterprise-wide consistency and usage of data at the derived level. Historical data at the reconciled level must therefore be as detailed and generic as possible. It must cover a time period at least equal to the longest required to manage the business.

As we see, historical data exists in all three layers, but for different reasons. The source of historical data is the real-time layer. Historical data is stored and sometimes used in this layer, but the main usage of historical data is in the derived layer for running the business. However, the role of historical data in the reconciled layer is very important because it builds the source for the derived layer.

2.3.2.3 Historical Data Volumes

The main characteristic of historical data is its potential volume and the associated costs of storage. However, this volume must be considered compared to its potential business benefits.

The time span required for the reconciled data layer can be deduced from the maximum time span needed at the derived data level. The consequence of not storing all detailed-level historical data is that some future historical analysis requirements will be impossible to meet, because some required data will no longer be available. This must be weighted against the cost of storing and managing all historical data. Sometimes, when defining the time span of historical data to be stored in the reconciled layer, it is better to store too much than too little, and define an active archival strategy to manage the data volumes.

CHAPTER 3

Online Analytical Processing

The new paradigm that OLAP (Online Analytical Processing) brings to us is the ability for the user to think of data logically as multidimensional. Business people, when they think analytically, think of their enterprises in dimensions: sales by region, profit by product, expenses by cost center, revenue by time, etc. Managing a business means tracking information in any number of dimensions.

OLAP is the process of creating and managing multidimensional enterprise data for analysis and viewing by the user who seeks an understanding of what the data is really saying. OLAP gives the user the tools to analyze the data and see what's good, what's bad, what's changed, and what's about to change by using comparisons to see the numbers in context. That means, with OLAP the user is not just navigating data, but exploring information.

This chapter introduces OLAP concepts and its relationship to data warehousing with focus on logical multidimensional features, such as dimensions, hypercubes, hierarchies, links, and formulas.

3.1 Functional Requirements of OLAP Systems

The requirements of OLAP systems share many of the standard requirements of any information system such as timely, accurate information. Beyond that, OLAP systems are unique in their attempt to provide the user with fast, flexible, friendly access to large amounts of derived data whose underlying inputs may be continuously changing. To accomplish these demanding goals, OLAP technology was forced to overcome the challenges of information overload that have arisen during the past couple of years. They include increases in the amount and complexity of data needed to make decisions, increases in the number of people who currently read and write to a common pool of data, increases in the amount of decentralized decision making, and increases in the distribution of data and processing that may pertain to a single query.

The importance of good information can be though of as the difference in value between right decisions and wrong decisions [Thomsen 1997]. The larger the difference between right and wrong decisions the greater the importance of having good information. For example, poor information about customer retail trends results in poor buying and allocation decisions for a retailer, which results in costly markdowns for what was overstocked and lost profit-making opportunity for what was understocked. Retailers tend to value accurate product-demand forecasts very highly. Good information about world events helps financial traders make better trading decisions, directly resulting in better profits for the trading firm. Major trading firms invest heavily in information technologies and good traders are rewarded.

Regardless of what information is being processed or how it is being processed, the goals are essentially the same. Good information needs to be accurate, timely, and understandable. The

first component of the functional requirements for OLAP comes from these general goals for good information processing: accuracy, timeliness, and understandability.

3.1.1 Differences between Operational Systems and Decision-oriented Systems

3.1.1.1 The Source of these Differences

Buying, selling, producing, and transporting are common examples of business operations. Monitoring, evaluating, comparing, budgeting, planning, projecting, and allocating resources strategically are common examples of business thinking that generates analysis based decision-oriented information.

The information produced through these thinking activities is analysis-based because some data-analysis, such as the calculation of a trend or a ratio or an aggregation, needs to occur as part of the activity. Knowing which products or customers are most profitable, or knowing which stores have sold the most this year is the kind of information needed in order to make decisions such as which products should have their production increased, or which customers should be targeted for special promotions, or which stores should be closed. The decision-orientation of analysis is essential. It serves to direct analysis toward useful purposes.

In contrast, many operational activities are decision-oriented without being based on analysis. For example, if a credit card customer asks for a credit increase, a decision needs to be made. If the customer is at his or her credit limit, the decision is no. The credit information was decision-oriented, but no analysis was involved in the decision.

Together, operations and decision-oriented analysis are at the core of all business activities, independent of their size, industry, legal form, or historical setting.

3.1.1.2 Current Differences between OLTP and Analysis-based Decision-oriented Processing

Operational software activities tend to happen at a relatively constant rate. Data is updated as frequently as it is read. The data represents a current snapshot of the way things are, and each query goes against a small amount of information. Operational queries tend to go against data that was directly input, and the nature of the queries is generally understood in advance, for example, retrieving a customer's address by pulling up his record knowing his account number.

In contrast to operations-oriented information activities, and on a less frequently basis, managers and analysts might ask higher-level analytical questions, such as what products have been most profitable for the company this year, is it the same group of products that were most profitable last year, how is the company doing this quarter versus the same quarter last year. The answer to these types of questions represents information that is both analysis-based and decision-oriented.

The volume of analysis-based decision-oriented software activities may fluctuate dramatically during the course of a typical day. On average, data is read more frequently than written. And when written, it tends to be in batch updates. Data represent current, past, and projected future states, and single operations frequently involve many pieces of information at once. Analysis

queries tend to go against derived data. And the nature of the queries is frequently not understood in advance. For example, a brand manager may begin an analytical session by querying for brand profitability by region. Each profitability number refers to the average of all products in the brand for all places in the region where the products are sold for the entire time period in question. Literally hundreds of thousands or millions of pieces of data may have been funneled into each profitability number. In this sense, the profitability numbers are high level and derived. If they had been planned numbers they might have still been high level, but directly entered instead of derived, so the level of atomicity for a datum is not synonymous with whether it is derived. If the profitability numbers looked unusual, the manager might then begin searching for why they were unusual. This process of unstructured exploration could take the manager to any corner of the database.

The differences between operational and analysis-based decision-oriented software activities are summarized in table 3.1 [Thomsen 1997].

Table 3.1: A comparison of operational and analysis-based decision-oriented information processing activities

Operational Activities	**Analysis-based Decision-oriented Activities**
More frequent	Less frequent
More predictable	Less predictable
Smaller amounts of data per query	Larger amounts of data per query
Query mostly raw data	Query mostly derived data
Require mostly current data	Require past, present data and projections
Few, if any, complex derivations	Many complex derivations

3.1.2 Requirements of OLAP Systems

End users are usually trying to get at more data and calculations based in that data, faster, and with greater viewing flexibility than can be achieved with traditional technologies, which are generally a combination of SQL databases, SQL-based data query tools, and spreadsheets.

Any software product claiming to fulfill the functional requirements for OLAP should provide fast, flexible, shared access to all analytical information. It should allow viewing and browsing through information any way the user likes. It should be powerful enough to calculate profits and allocations across products, divisions, and currencies. It should be friendly enough to be learned by a non-technical person with a minimum of effort. And it should be integrated in both a user sense – integrating multiple users by allowing them simultaneous access to the same data – and a data sense – integrating data from across the enterprise and its environs.

1. Fast Access and Calculations
   OLAP requires supporting ad hoc queries, some of which may require computations performed on the fly. For example, someone might start a session by querying how overall product profitability was in Europe last quarter. Seeing that profitability was lower than expected, he or she might navigate down into individual countries while still looking at overall product profitability. Here a person might see that some countries were significantly below the others and so would naturally navigate further down into product groups for these countries, always looking for some data that could

help explain the higher-level anomalies. Here he or she might find that it was not the sales that was unusual, but rather the indirect costs were substantially higher in for these countries. Since each of this steps constitutes a query, the goal of OLAP systems is to provide fast response time regardless of the type of query or the size of the database.

2. Powerful Analytical capabilities
   As most of the important information results from the intelligent comparison of ratios and inferred trends over time and other dimensions, a good part of the querying that takes place in analytical processing contains embedded analysis. A sales director might want to know which product categories had profit levels abroad that differed the most from profit levels in his country. He might also want to see the results ordered from most positive to most negative. Therefore, the system needs to perform a variety of calculations. Profits need to be calculated by product. Profit values need to be normalized so they can be compared across products. Normalized profit levels need to be aggregated to the product category level. This needs to be done across time for different countries in different time zones. Time aggregations may need to be adjusted for differences in reporting periods and/or number of days per period. The profit level for each product group then needs to be compared between the two markets, and finally the differences in profit level need to be ordered from most positive to most negative. In an OLAP system, these types of calculations should be as straightforward as they are to say.

3. Flexibility
   OLAP systems need to be flexible in many ways. This includes flexible viewing, flexible definitions, flexible analysis, flexible interfaces. They need to support a full range of unplanned calculations because analysis-based decision-oriented thinking is difficult to automate or specify. View flexibility means that the user can easily choose to see information in the form of graphs, matrices, or charts, and within any form, such as matrices, and in any orientation of row and column headers. In terms of definitions, users should be able to change the names of descriptors, the formatting of numbers, the definitions of formulas, the triggers for automatic processes, and the location of source data.

4. Multiuser Support
   As a result of downsizing and decentralizing of organizations, the relative number of employees who need read and write access to decision-support analytical data is on the rise. Problems discovered by a regional sales manager, for example, may need to be communicated to a distribution or manufacturing manager for prompt resolution. Forecasts examined by senior executives may reflect data that was generated from many different departments. For global corporations, some of those departments may not even share the same country or language. OLAP systems should provide multiuse read-only or read/write access to the information.

As we can see, that information processing is part of all organizations regardless of the type of business they do. The major subdivisions in information processing reflect the major subdivisions of organizational activity: operations and decision-oriented analysis. The terms OLAP and data warehousing are complementary terms, each of which refers to a component of the overall functional requirements of analysis-based decision-oriented information processing. Whereas data warehousing represent a server-centric or supply-side view of

analysis-based decision-oriented information processing, OLAP represents the use-centric view of the same function. The major requirements for OLAP are fast data access, fast calculations, computational expressiveness, user-friendly interfaces, flexible views, and multiuser support.

3.2 Multidimensional Features

Multidimensional software is specifically designed to facilitate the definition and computation of sophisticated multilevel aggregations and analysis. The major features of any multidimensional software product or tool include hypercubes, dimensions, hierarchies, links, and formulas. This section provides a quick overview of the key multidimensional features and their problem-solving benefits.

3.2.1 Dimensions, Hierarchies and Hypercubes

The first key feature of any multidimensional tool is the ability to define a data set in terms of multiple dimensions. Generally speaking, dimensions may be though of as major perspectives, entities, factors, or components. For example, the major dimension in a sales tracking system might be time, location, salesperson, customer, and product. The major dimension in a loan application might be time, branch, customer, and loan type. In database terms, a good analogy for a dimension is a key. So a multidimensional data set may be though of as a multikey or multiattribute data set. The benefit of handling multiple dimensions is in being able to represent the full richness of a data set in a single model or cube.

A multidimensional system must be able to display the model dimensions in any three-dimensional grid configuration – consisting of rows, columns, and pages – on a computer screen. In other words, any model dimensions such as time, store, customer, and product can be shown in any row by column by page screen configuration. This is extremely useful for all kinds of ad hoc querying and analysis, especially when the number of dimensions exceeds two or three.

The ability to easily change views of the same data by reconfiguring how dimensions are displayed is one of the great benefits of multidimensional systems. It is due to the separation of data structure from data display. As distinguished from physical dimensions, which are based on angles and limited to three, logical dimensions have no such limit.

A very important key feature of any multidimensional system is the hierarchical nature of the dimensions. Any dimension, such as time, product, and store can have multiple levels of granularity. For example, the time dimension may have a day-level granularity, a week-level granularity, a month-level granularity, a day-level granularity, and so on. This is indispensable for working with larger data sets that invariably need to be aggregated, analyzed, and viewed across multiple levels of aggregation.

Hierarchies are the foundation for aggregating data and for navigating between levels of detail within a hypercube. Relative referencing within a hierarchical environment is more complicated than within a row and column environment as the former is direction specific. Although hierarchies are not a necessary part of any dimension, all real-world applications of

moderate or greater complexity involve some hierarchical dimensions, such as time, geography, product, customer, or market.

The combination of multiple dimensions and multiple levels per dimension constitutes the essence of a multidimensional cube or hypercube. A cell in a hypercube is defined by the intersection of one member from each dimension. The more dimensions and hierarchies are in the cube, the more complex is the neighborhood surrounding any cell. In an N-dimensional hypercube (with one hierarchy level per dimension), each cell has 2N immediate neighbors (an immediate neighbor to a cell differs from that cell by one unit of one dimension).

The term "drill down" refers to the process of navigating either directionally or by endpoints toward greater detail. The greater detail can come from moving down along any dimension. The term "drill up" is simply the reverse of drill down.

3.2.2 Data

Although the majority of data residing in real-word implemented hypercubes in numeric, any kind of media from text to graphics, and even sound, may be multidimensional. The key issues related to data and whether they belong in a hypercube are as follows:

- Identifying the value of bringing the data into a hypercube
- If the value is there, finding a tool that can do the job

Beyond numbers, many tools provide the ability to populate hypercubes with textual data. The two main ways that an OLAP tool adds value to data is through organization and aggregation (in multiple levels and dimensions). Numeric data is so well suited for OLAP applications because it has a dimensional organization and because it is easy to aggregate. Other data types may benefit from a dimensional organization, but the issues surrounding their aggregation go well beyond the numeric computation abilities of today's OLAP tools.

Most of the corporate data held in databases around the world is character- or text-string-based. For this reason, character string data have become increasingly important for OLAP tools to handle properly. Character string data like color, address, package type, customer type are essential factors in analysis. It is often required to analyze whether product sales are a function of package type or color, or whether there is any correlation between the type of complaints a chain receives and the location of the store where it was made.

Numeric and nonnumeric data is not always cell-based, but may also attach to the members of a dimension. This kind of data is called "attribute data". For example, a store dimension may contain information about the address, phone number, and square footage of each store. These pieces of information would be called attributes of the store dimension.

From a data warehouse perspective, most of the attribute data exists in the dimension tables and can be just as important as cell-based fact data for the purpose of analyzing a hypercube. For example, it may be required to see sales broken down by stores grouped according to the attributes square footage, or number of floors, or the existence of parking, or its opening date. Dimensions may have dozens of attributes. Most multidimensional tools provide a way to store and analyze attribute information.

3.2.3 Links

Links between separate multidimensional and operation systems define the method for maintaining a persistent bidirectional connection between the hypercube and a changing external data set. OLAP tools need to have the ability to establish persistent links with external sources of data, such that when changes occur in the external data sources, they are automatically brought into the hypercube. And any dimension modifications or data calculations are automatically and incrementally performed.

Because OLAP products and implementations are generally separate from the systems that generate the date to be manipulated and analyzed, links serve essentially as transformation functions. Links can be read only or read/write, static or dynamic. Static links are not capable of processing changes made to the source. They are used only by the multidimensional toll when it loads or refreshes its information from the source. Dynamic links maintain an open connection between the source and the hypercube wherein changes to the source are propagated to the hypercube.

There are three basic types of links: structure links, attribute links, and content links. A structure link is used to extract structural information for a dimension, identifying the members and their referential or hierarchical structure. An attribute link is used to map attribute information to the members of a dimension. Content links are used to map data into the hypercube.

3.2.4 Formulas

Another key feature of any multidimensional system is the ability to attach formulas to members of dimensions. Because a single member of one dimension, for instance the "sales" member from a variables dimension, interacts with every single member from every other dimension, a single-dimensional formula has a powerful application range frequently doing the same work that would otherwise take thousands of spreadsheet formulas. Also, unlike SQL formulas, multidimensional formulas work equally well in all dimensions. Multidimensional formulas dramatically simplify the process of defining aggregation and analysis calculations.

Formulas are used for aggregating, allocating, analyzing, explaining, and inferring. For example, net sales may be defined by a formula such as gross sales minus returns, and business products may be defined as the sum of computer products, fax machines, and photocopiers. Next year's projected sales may be defined as the average sales growth multiplied by the current sales.

Instead of being defined in terms of individual cells, multidimensional formulas are defined in terms of members of dimensions, which means they apply to all cells in the cube sharing that particular member. As the number of cells to which a formula may apply can be very large, it is common for member formulas to be conditional upon the values of members in other dimensions. Because there can exist one formula per member per dimension, frequently more than one formula applies to a particular cell. In these cases a precedence rule needs to be invoked to determine which formula (or ordering of formulas) will be evaluated for the cell. The combination of multiple dimensions, flexible screen representations, multilevel dimensions, and dimension formulas represent the core of any multidimensional system.

3.3 Benefits of the Multidimensional Approach

Typical business models require multiple levels of data aggregation across multiple dimensions. End-user analysts need to be able to browse the data while changing the configuration of its display in the screen. And they need to be able to analyze data, looking most often at comparisons along dimensions.

A multidimensional approach offers many clear advantages over spreadsheets and SQL for both defining and using such models. The separation of data structures (defined in terms of dimensions) from data representation is a big advantage of the multidimensional approach. It serves to minimize the need to duplicate any structural information and it provides direct support for easily changing views on a screen. The direct support for multilevel dimensions, and the ability to assign formulas directly to the members of dimensions, makes it easy to define multilevel aggregates and multidimensional calculations with a multidimensional tool.

Multidimensional tools provide the ability to define data in terms of dimensions. For example, the major dimensions in a sales tracking system might be time, region, product and customer. The benefit of using multiple dimensions is being able to represent the complete data set in a single model called hypercube. The ability to display the model dimensions in any three-dimensional grid configuration on a computer screen is very useful for all kinds of ad hoc queries and analyses.

For more information about OLAP concepts and features please refer to [Thomsen 1997].

3.4 The Codd Rules and Features

In 1993, Codd and Date introduced 12 rules covering the field of OLAP. [Codd 1993] defines OLAP as "the dynamic synthesis, analysis, and consolidation of large volumes of multidimensional data".

With OLAP, Codd and Date has addressed an important area of processing business data and a new category of products for "multidimensional analysis", which are not well covered by the combination of relational systems, query tools, and spreadsheets.

[Codd 1993] defines OLAP as "the name given to the dynamic enterprise analysis required to create, manipulate, animate and synthesize information from Enterprise Data Models. This includes the ability to discern new or unanticipated relationships between variables, the ability to identify the parameters necessary to handle large amounts of data, to create an unlimited number of dimensions (consolidation paths) and to specify cross-dimensional conditions and expressions."

"OLAP involves the dynamic and extensive manipulation of unlimited variables of data, and complements OLTP applications. OLAP applications include rapid consolidations and multiple scenarios of forecasts, budgets, product plans, capital asset plans, sales analyses, and performance reports. OLAP products make multidimensional analysis easier and more efficient for business users".

The twelve rules of Codd to evaluate OLAP products are:

1. Multidimensional Conceptual View
2. Transparency
3. Accessibility
4. Consistent Reporting Performance
5. Client-Server Architecture
6. Generic Dimensionality
7. Dynamic Sparse Matrix Handling
8. Multi-User Support
9. Unrestricted Cross-dimensional Operations
10. Intuitive Data Manipulation
11. Flexible Reporting
12. Unlimited Dimensions and Aggregation Levels

1. *Multidimensional Conceptual View*
   The view of a user analyst of the enterprise is typically is multidimensional. Thus, the conceptual view of the user analyst of OLAP models should be multidimensional as well. This multidimensional conceptual schema or user view makes model design and analysis possible, but also facilitates dimensional calculations through the analytical model. This way, it is easier for user analysts to manipulate these multidimensional data models than single dimensional models. For example, multidimensional models make many manipulations, which take more time and effort with older approaches, easier , such as slice and dice, or pivot and rotate consolidation paths within the model.

2. *Transparency*
   For the user, it should be transparent whether OLAP is part of the user's front-end product (e.g., spreadsheet or graphics package) or not. If OLAP is offered as a client-server architecture, then this fact should also be transparent to the user analyst. It is preferred to offer OLAP as a true open systems architecture, and thus to allow the user-analyst embedding the analytical tool anywhere, without affecting the functionality of the host tool.

   Transparency is critical to preserving the user's proficiency and productivity with the existing front-end, providing the appropriate function level, and assuring that no complexity is added in any way. Besides, the source of the enterprise data eneterd to the OLAP tool (i.e. whether it is a homogeneous or heterogeneous database environment) should be transparent to the user as well.

3. *Accessibility*
   It should be possible for the OLAP user analyst to perform analysis using a predefined conceptual schema which is composed of enterprise data in relational DBMS, but also data residing from legacy DBMS and other data stores. This should be the basis of a common analytical model.

   This requires the OLAP tool to map the logical schema to heterogeneous physical data stores, access the data, and perform the needed conversions to provide a single,

consistent and coherent view to the user. Furthermore, not the user analyst, but the tool must be able to deal with the source the physical data and its system type. It is important that the OLAP system accesses only the required data to perform the needed analysis.

4. *Consistent Reporting Performance*
With the number of dimensions or the database size increasing, there shouldn't be any significant degradation in the performance of reporting. Reporting performance should be consistent to assure user-friendliness and reduce complexity when bringing OLAP to the end user.

5. *Client-Server Architecture*
Data that requires OLAP is typically stored on mainframe systems and accessed using personal computers. Therefore, it is necessary that OLAP tools can operate in a client-server environment. The server of OLAP tools must be intelligent enough so that it is possible to attach several clients with little integration and programming.

It is also important that the server can perform the mapping and consolidation between logical and physical enterprise database schema. This has a significant impact on transparency and facilitates building a common conceptual, logical and physical schema.

6. *Generic Dimensionality*
Data dimensions have to be all equivalent structure and operational capabilities. Selected dimensions may be granted additional operational capabilities, but as dimensions are symmetric, any additional function could be granted to any dimension. The basic data structure, formulas, and reporting formats should be neutral toward all data dimensions.

7. *Dynamic Sparse Matrix Handling*
The physical schema of OLAP tools has to fully adapt to the specific analytical model, which provides optimal sparse matrix handling. There is one and only one optimum physical schema for any given sparse matrix. Unless the complete data set can be cached in memory, this schema provides matrix operability and maximum memory efficiency.

The basic physical data unit of the OLAP tool has to be configurable to any subset of the existing dimensions, in any order, to provide practical operations within large analytical models. Moreover, it should be possible to dynamically change the physical access methods, which should contain different types of mechanisms like:

- direct calculation;
- B-trees and derivatives,
- hashing;
- the ability of combining these techniques if needed.

Sparseness is only of the characteristics of data distribution. Fast, efficient operation can be unobtainable if it isn't possible to adjust to the data distribution of the data set. Models appearing to be practical, based on the number of consolidation paths and dimensions, or the size of the enterprise source data, may be too large or too slow in

actuality if the OLAP tool is not able to adjust according to the distribution of values of the data that needs to be analyzed. Regardless of the order of cell access, access speed should always be consistent and stay constant across models with different numbers of data dimensions or different sizes of data sets.

For instance, if the set of input data from the enterprise database is very dense, predicting the size of the resulting data set is easily possible after consolidation across all modeled data dimensions. In a five-dimensional analytical model, it can be supposed that after model consolidation the physical schema size is two-and-a-half times the size of the input data from the enterprise database.

Nevertheless, the physical schema could be two-hundred times the size of the enterprise data input if the enterprise data is very sparse and is distributed based on certain characteristics. But, if the data set size is the same, and the sparseness degree is the same, but the data distribution is different, the size of the physical schema could be much smaller, especially if the input data is very dense.

Statistical analysis tools typically compare two dimensions against each other, regardless of other data dimensions. That makes them unsuitable to multidimensional data analysis. Even if these tools could compare all dimensions to each other at the same time, this would result in the size of the product of all the data dimensions, which would be the maximum size of the physical schema.

OLAP tools give user analysts the ability to perform types of complex analysis by adapting the physical data schema to the existing analytical model. Because the behavior of multidimensional data models is extremely unpredictable and volatile, it is not possible to successfully use tools with a static physical schema and a data storage unit with a fixed number of dimensions.

A physical schema with fixed dimensionality can be optimal for one analytical model but impractical for most others. OLAP tools have to adapt the model's physical schema dynamically to the indicated dimensionality and to the data distribution of each model instead of basing a physical schema upon cells, records, two dimensional sheets, etc.

8. *Multi-User Support*
   Several user analysts often need to work simultaneously with the same analytical model or to create various analytical models from the same enterprise data. Therefore, OLAP tools have to provide simultaneous access to the analytical model.

9. *Unrestricted Cross-Dimensional Operations*
   The different roll-up levels represent the most of 1:1, 1:N, and dependent relationships in an OLAP model or application. This is because of the inherent hierarchical nature they have. Therefore, the OLAP tool should not expect the user analyst to define these calculations, but rather they are supposed to provide the associated calculations.

   Calculations which do not result from these relationships need different formulas to be defined according to some computationally complete language. This language must not inhibit or restrict any relationship between data cells, but rather offer calculation and data manipulation across any number of dimensions with different number of

common data attributes.

For instance, let's look at the difference between a single dimensional and a cross-dimensional calculation. In a single dimensional a calculation like *Revenue - Cost = Contribution* specifies a relationship between attributes in a single dimension, for example *ACCOUNT_DIM*. For all cells of all data dimensions in the data model that contains the attribute *Contribution* Upon calculation, the relationship is calculated. The calculations of a cross-dimensional relationship may cause some additional challenges. For instance, consider this five-dimensional structure:

```
Account_Dim
  Sales
  InterestRate
  Overhead
  etc

Corporate_Dim
  United States
          Washington
          New York
          etc
  Canada
          Montreal
          Ottawa
          etc

FiscalYear_Dim
  Quarter1
          January
          February
          March
  Quarter2
          April
          May
          June
  etc

Products_Dim
  Electronics
  Books
  etc

Scenario_Dim
  Actual
  Budgeted
  Variance
  etc
```

To allocate corporate overhead to parts of the company like local offices (Montreal, Ottawa, etc) the formula using their respective contributions to overall sales could be like this:

Overhead is the percentage of total sales represented by the sales of each single local office multiplied by total company overhead.

Another example shows the need for cross-dimensional calculations. The user analyst may require that for all Canadian cities, the variable *InterestRate* that is used in other calculations, is to be set to the value of the *Budgeted January Interest rate* for the city of Montreal for all months, across all data dimensions.

If the user analyst had not specified the city, scenario, and month, the attributes would change and remain consistent with the month attributes of the data cell being calculated in the analytical model. The required calculation might look like this:

If the value within the examined cell appears in the path Corporate_Dim, underneath the level Canada, ***then*** *the global interest rate is set to the value of the interest rate for the month of January which is budgeted for the city of Montreal.*

10. *Intuitive Data Manipulation*
    Re-orientation of a consolidation path, drilling down rows or across columns, zooming in and out, etc. need to be performed using direct action upon the analytical model cells. They should not need a menu or several steps via the user interface. The dimensional view of the user analyst should provide all the required information which affects these actions.

11. *Flexible Reporting*
    Data analysis and presentation is easier when columns, rows, and cells of data needing to be compared are grouped logically in the enterprise. Data that needs to be synthesized or information that comes from changes to the data model according to any kind of orientation should be presented by reporting functionalities. This requires the columns, rows, or page headings to be able to display any number of dimensions in the analytical model.

    Besides, each dimension must be able to display any subset of the members, in any order, and show the inter-consolidation path relationships between the members of this subset.

12. *Unlimited Dimensions and Aggregation Levels*
    The number of dimensions analytical models may require may be up to nineteen or even more concurrent data dimensions. Therefore, it is strongly recommended that an OLAP tool should be able to handle at least fifteen if not twenty data dimensions within a common analytical model.

    Moreover, each of these dimensions has to allow any number of aggregation levels defined by the user analyst within any consolidation path.

These rules were followed by another six rules in 1995:

13. Batch Extraction vs. Interpretive
14. OLAP Analysis Models
15. Treatment of Non-Normalized Data
16. Storing OLAP Results: Keeping Them Separate from Source Data
17. Extraction of Missing Values
18. Treatment of Missing Values

13. *Batch Extraction vs. Interpretive*
    OLAP products are required to offer their own staging database for OLAP data, but also live access to external data. Although this is a very effective feature, only a few OLAP products actually complies with it, and still those products often do not make it easy or automatic. Therefore, Codd was recommending multidimensional data staging with partial pre-calculation of large multidimensional databases that goes to the detail.

14. *OLAP Analysis Models*
    OLAP products should support four models of analysis as described by Codd. These include categorical, exegetical, contemplative and formulaic models.

15. *Treatment of Non-Normalized Data*
    This rule addresses the integration between an OLAP engine and denormalized source data. Codd required that "any data updates performed in the OLAP environment should not be allowed to alter stored denormalized data in source systems". This could also be interpreted as saying that "data changes should not be allowed in what are normally regarded as calculated cells within the OLAP database".

16. *Storing OLAP Results: Keeping Them Separate from Source Data*
    This feature can be considered as an implementation issue rather than a product issue. It requires that read-write OLAP applications should not be implemented directly on operational data, and OLAP data changes should be kept separated from operational data. A good example of such an implementation is the method of data write-back used in Microsoft Analysis Services because it keeps the effects of data changes even within the OLAP environment separated from the base data.

17. *Extraction of Missing Values*
    Missing values should not be distinguished from zero values. However, to facilitate storing sparse data more compactly, some OLAP tools tend to break this rule, but this doesn't cause a great loss of functionality.

18. *Treatment of Missing Values*
    Regardless of their source, all missing values should be ignored by the OLAP analyzer. This is an almost inevitable consequence of the way in which multidimensional engines treat all data.

For more details please refer to [Codd 1993].

CHAPTER 4

Representing Time in the Data Warehouse

4.1 Representing Time in Business Data

4.1.1 The Need for Timestamps

As Business tends to change over time, the business data must be able to represent that change. However, today most data modeling and application design approaches focus only on a static view of the world.

Let's take as an example a common entity relationship ER model and look at the relationship between two entities: employee and department as shown in Figure 4.1. A department can have zero to many employees, and an employee belongs to one and only one department. Thus the cardinality of this relationship is 1:n. This is very important to know as we move from the logical model to the physical implementation of the database.

Figure 4.1: Cardinalities in the ER model

However, the statement "an employee belongs to one and only one department" is true only at one point in time, simply because an employee can change his department and thus belong to many departments at different times. This change is completely ignored by the ER model.

As events can take place to change the relationship between two entities, in the last example an employee and his department, those events somehow need to be considered and represented in the data model as well. But since today's modeling tools and databases provide little or no support for time dependence the designers try to add time dependence of data to the application design.

This might be adequate in operational applications, which manage only real-time data and take a view mainly of the current state of the business. A data warehouse, however, must explicitly consider the temporal aspects of the data it contains, because it must, by definition, provide an historical view of the business.

One important approach that is widely used is adding timestamps to the data. A timestamp is a specially defined field, in data-and-time format, that tracks when a data record has been created, deleted, or changed in any way. Changes can be tracked on field-, record/row, or file/table level, depending of the required level of detail and the available storage.

4.1.2 How Data is Stored

In addition to representing time in a database, changes in data have to be captured and represented over time. These changes are the result of the occurrence of individual events in the database, which cause a change of the status of the stored data. It is possible to store either events or statuses, and the informational environment may require both approaches.

Status database: A database containing point-in-time records showing the state of an entity after the occurrence of an event.

Event database: A database containing a record of the events that cause the values of an entity to change.

While status data is the normal day-to-day data stored in databases and files for many operational and informational applications, event data is often not stored for business purposes at all. Event data is stored in database logs for recovery purposes, even though the data structure in such logs is too complex for general use. An important use of event data is to support data replication.

When we compare the two approaches, we can see that the status approach stores larger volumes of data because in any change most of the fields remain unchanged, but are duplicated anyway. In the event approach, only the primary record key and the changed fields need to be stored at each change, and thus the amount of data stored here is far less than the complete record. However, if the data structure is normalized, the length of the status data record decreases, and also the difference in the storage volume required by the two approaches decreases. The flexibility of the event approach sometimes results in both approaches -or a combination of both- being used.

For example, customer data would be stored as status records because it changes only infrequently, while bank account data would store individual events, which provides more flexibility when taking different views on the data.

In both cases it should be possible to convert from one view to another if required by the business. However, converting stored statuses into events is generally simpler than the other way, as all it requires is taking records in pairs and calculating the difference. On the other hand, converting events into statuses means applying all the events in sequence from the initial one until the required time, which may take a long time.

4.1.3 Temporal Data Structures

Timestamps and the concepts of status or event representation allow the maintenance of temporal data. But there is one more aspect of temporal data to understand how history is reflected in a database. This relates the structure of the data and how new events affect existing data. This leads to the definition of three data structures:

Transient data is real-time data in which changes to existing records overwrite the previous data, and deletions physically erase records, leading to a loss of the historical record of the changes that data has. This type of data is found in the real-time data of the operational environment and is only stored as statuses. Because a stored status is replaced completely by

the changing event, each status record is only available until the next event that changes this record occurs.

Periodic data is data recording the history of a business over a period of time by maintaining a complete record either of all statuses or of all events that have occurred. Once a record is added, it is never physically deleted, nor is its business content ever physically modified. Rather, new records are always added, even for updates to or deletions of existing records. Periodic data is persistent in nature because it provides a complete record of the data and its changes. Either statuses or events can form the basis for this complete record.

Periodic data is found in the real-time data of the operational environment where a record of the previous statuses or data is required, e.g. bank account and insurance systems. However, due to performance and storage issues, sometimes the duration to hold this persistent data is short. If this is the case then we are talking about semi-periodic data.

Snapshot data is a point-in-time view of the business data showing its status at a particular time, which is then kept as a permanent record of that state of the data. Snapshots usually represent the business data at some time in the past, and a series of snapshots can provide a view of the history of the business. However, predictions and plans can also be captured as snapshots, in this case representing the future. Snapshot data is also persistent in nature.

4.2 Temporal Issues

4.2.1 The Time Dimension

The TIMESTAMP type available in SQL provides a representation of time in a very fine precision. It stores the year, month, day, together with the hour, minute, second, and number of fractional digits of the second. It can be used in a database table to record the occurrence of certain events (e.g. deposit to and withdrawal from a bank account), as well as the start and end of a certain state (e.g. a certain employee belongs to a certain department). See Figures 4.2 and 4.3.

account_no	transaction_time	amount
326725348	13-MAR-2003 16:32:28	500
748746887	21-MAR-2003 10:13:45	350
848764836	21-MAR-2003 13:06:25	980

Figure 4.2: Deposits table using timestamps

emp_no	dept_no	start_date	end_date
130	30	01-APR-1998	30-SEP-2001
130	35	01-OCT-2001	31-DEC-9999
125	30	01-JUL-2002	31-DEC-9999

Figure 4.3: Employment table using timestamps

As we move toward a multidimensional approach the simple timestamp is replaced with a time dimension. The time dimension is then filled with a lot of helpful calendar attributes and is connected to the fact table by a foreign key (Figure 4.4).

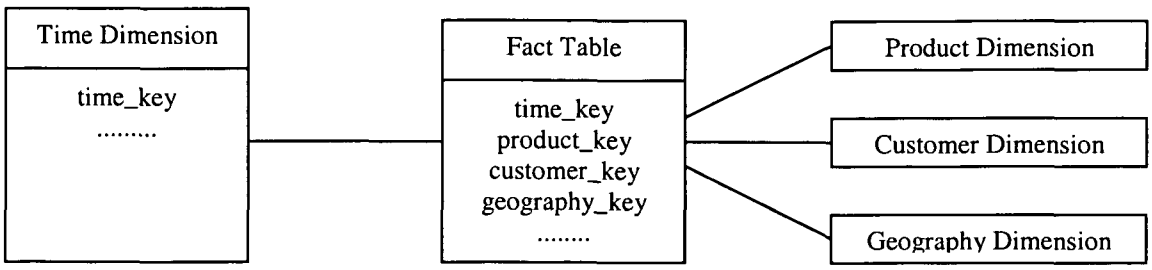


Figure 4.4: A multidimensional model

In SQL the time dimension can be created like this:

```
create table time_dimension (
   time_key                integer primary key,
   sql_date                date not null,
   day_of_week             varchar(9) not null, -- 'Sunday', 'Monday'...
   day_number_in_month     integer not null, -- 1 to 31
   day_number_overall      integer not null, -- first day is 1
   week_number_in_year     integer not null, -- 1 to 52
   week_number_overall     integer not null, -- weeks start on Sunday
   month                   integer not null, -- 1 to 12
   month_number_overall    integer not null, -- first month is 1
   quarter                 integer not null, -- 1 to 4
   year                    integer not null
);
```

But do we really need a dimension for time? Wouldn't it be better to use an SQL timestamp in the fact table instead of the foreign key and avoid this expensive join? To answer this question let's take a look at these simple queries:

- Show all the transactions that occurred within a given period of time.
- Determine whether a certain transaction occurred within a given period of time.
- Show transactions using complex calendar navigation capabilities including seasons, fiscal periods, day numbers, week numbers, weekdays and holidays.

While the first two queries are pretty simple to define using a single timestamp that stores the occurrence time of each transaction, this is not the case for the third query. Since SQL timestamp know nothing about an organization's calendar, fiscal periods or holidays, these attributes are modeled using a time dimension (Figure 4.5). This way the application designer doesn't have to embed these calendar constrains in the application design, which would require a set of complex queries to determine these attributes. Besides that this would be very slow, the end-user application can't easily produce the needed SQL.

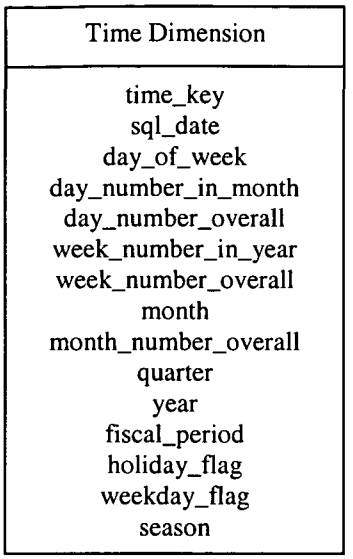


Figure 4.5: The time dimension

A time dimension can easily be built using a simple spreadsheet. A 20-year time dimension on daily basis contains about 7300 rows, which is not much. It can also be filled with a single SQL INSERT statement, as we will see later. However, problems will start to arise when the fact table requires granularity smaller than a day, let it be an hour, a minute, or a second. It's not possible to create a single time dimension with all the minutes or seconds over a long period of time. There are more than 500,000 minutes and 31 million seconds in a year. So, for these cases the only way is to use SQL timestamps despite the limitations we mentioned and to give up the ability to navigate through seasons and fiscal periods to the nearest second. We will come to this later when we talk about granularity to see how to overcome this issue.

4.2.2 Holidays, Seasons and Fiscal Periods

The time dimension as defined above gives us the ability to track business facts very well on a daily basis. But as the business requirements become more complex the time dimension must be extended by new attributes to meet those requirements. For example, the business might require looking at sales on holidays versus non-holidays. With an OLTP model holidays are usually stored in an own table, which can look like this:

```
create table holidays (
holiday_date              date primary key
);
```

This table is filled with all the holidays and can be joined with the sales table by date key. Thus a query is then used to join this table with the sales table:

```
select sum(sales.quantity_sold)
from sales, holidays
where trunc(sales.date_time_of_sale) = trunc(holidays.holiday_date);
```

As this looks pretty simple, since the holidays table eliminates sales on days that are not holidays, the case looks slightly different when we want to eliminate sales on days that are holidays. The query will look like that:

```
select sum(sales.quantity_sold)
from sales
where trunc(sales.date_time_of_sale)
  not in (select holiday_date from holidays);
```

Of course this query will take longer time to execute. The sales table may contain millions of rows, and the holidays table will contain about 50 to 100 rows. Here, the sub-query will be performed for each row examined by the main query. So the time to execute this query might be much longer than the holidays query. Besides, it will become more complex when we want to look at sales, not just on holidays, but also on different seasons, fiscal periods or weekdays. For each of these attributes a separate table is necessary, which then must be joined with the sales table.

Here comes the big advantage of the time dimension. All these attributes can be integrated into the time dimension in a way that effectively reduces query execution time and provides more functionality than using conventional RDBMS tables. The tables for holidays, seasons, fiscal periods and weekdays are replaced with attributes and flags in the time dimension, which will look like this:

```
create table time_dimension (
   time_key                integer primary key,
   -- this is midnight (TRUNC) of the date in question
   sql_date                date not null,
   day_of_week             varchar(9) not null, -- 'Sunday', 'Monday'...
   day_number_in_month     integer not null, -- 1 to 31
   day_number_overall      integer not null, -- first day is 1
   week_number_in_year     integer not null, -- 1 to 52
   week_number_overall     integer not null, -- weeks start on Sunday
   month                   integer not null, -- 1 to 12
   month_number_overall    integer not null, -- first month is 1
   quarter                 integer not null, -- 1 to 4
   year                    integer not null,
   fiscal_period           varchar(10),
   holiday_flag            char(1) default 'f'
                           check (holiday_flag in ('t', 'f')),
   weekday_flag            char(1) default 'f'
                           check (weekday_flag in ('t', 'f')),
   season                  varchar(50),
);
```

Now if we define our fact table to have the `time_key` as a reference to the time dimension, there will be no need to store an SQL date or timestamp in the fact table. The time dimension stores, for each day, the following attributes:

- whether or not this day is a holiday
- into which fiscal period this days falls
- which season is this day part of
- the day of week for this day

Of course these attributes must be defined and filled according to the organization's fiscal calendar. Now if we want to report sales by season, the query will be straightforward:

```
select td.season, sum(f.dollar_sales)
from sales_fact f, time_dimension td
where f.time_key = td.time_key
group by td.season;
```

Using the `group by` command we can report by other attributes like holidays and fiscal periods in an identical way, which makes the queries much faster than using separate tables. Using other dimensions like products, customers, manufacturers, etc. makes us able to report by different criteria.

As mentioned above, the time dimension can be populated using a spreadsheet or even easier with a simple SQL INSERT statement. This is done using SQL date formatting functions and a help table `integers`, which supply a series of numbers to be added to a selected starting date.

For this example let January 1st 2000 be the first date:

```
-- The insertion is driven by the use of the integers table,
-- which just contains a set of integers, from 0 to n.
-- d is the SQL date of the day we're inserting.

insert into time_dimension
  (time_key, sql_date, day_of_week, day_number_in_month,
   day_number_overall, week_number_in_year, week_number_overall,
   month, month_number_overall, quarter, year, weekday_flag)
  select n, d, rtrim(to_char(d, 'Day')), to_char(d, 'DD'), n + 1,
       to_char(d, 'WW'), trunc((n + 6) / 7),
        -- January 1, 2000 was a Thursday, so +6 to get s
        -- the week number to line up with the week
       to_char(d, 'MM'), trunc(months_between(d, '2000-01-01') + 1),
       to_char(d, 'Q'), to_char(d, 'YYYY'),
       decode(to_char(d, 'D'), '1', 'f', '7', 'f', 't')
  from (select n, to_date('1998-01-01', 'YYYY-MM-DD') + n as d
      from integers);
```

The remaining fields (`season`, `fiscal_period`, `holiday_flag`) cannot be filled using SQL date functions and have to be populated afterwards. Fiscal period and season depend on the organization's choice of fiscal year. To update the `holiday_flag` field, which is `'f'` by default, we need two help tables: one for the fixed holidays and one for the floating holidays.

```
create table fixed_holidays (
   month          integer not null check (month >= 1 and month <= 12),
   day            integer not null check (day >= 1 and day <= 31),
   name           varchar(100) not null,
   primary key (month, day)
);
```

```
-- Specifies holidays that fall on the n-th DAY_OF_WEEK in MONTH.
-- Negative means count backwards from the end.

create table floating_holidays (
   month          integer not null check (month >= 1 and month <= 12),
   day_of_week    varchar(9) not null,
   nth            integer not null,
   name           varchar(100) not null,
   primary key (month, day_of_week, nth)
);
```

Some example holidays:

```
insert into fixed_holidays (name, month, day)
   values ('New Year''s Day', 1, 1);
insert into fixed_holidays (name, month, day)
   values ('Christmas', 12, 25);

insert into floating_holidays (month, day_of_week, nth, name)
   values (11, 'Thursday', 4, 'Thanksgiving');
insert into floating_holidays (month, day_of_week, nth, name)
   values (5, 'Monday', -1, 'Memorial Day');
```

After that, it is easy to update the `holiday_flag` in the time dimension using these two help tables. The following pseudocode can be implemented using any procedural language like PL/SQL to set the `holiday_flag` to `'t'` for the days just inserted into the two tables.

```
for row in "select name, month, day from fixed_holidays"
    update time_dimension
      set holiday_flag = 't'
      where month = row.month and day_number_in_month = row.day;
end for

for row in "select month, day_of_week, nth, name from
floating_holidays"
    if row.nth > 0 then
   -- If nth is positive, put together a date range constraint
   -- to pick out the right week.
        ending_day_of_month := row.nth * 7
        starting_day_of_month := ending_day_of_month - 6

   update time_dimension
          set holiday_flag = 't'
          where month = row.month
            and day_of_week = row.day_of_week
            and starting_day_of_month <= day_number_in_month
            and day_number_in_month <= ending_day_of_month;
    else
   -- If it is negative, get all the available dates
   -- and get the nth one from the end.
        i := 0;
        for row2 in "select day_number_in_month from time_dimension
                        where month = row.month
                          and day_of_week = row.day_of_week
                        order by day_number_in_month desc"
            i := i - 1;
            if i = row.nth then
                update time_dimension
                  set holiday_flag = 't'
```

```
                    where month = row.month
                      and day_number_in_month = row2.day_number_in_month
                  break;
              end if
          end for
      end if
  end for
```

In order to consider holidays in different countries or in different time zones we could use multiple holiday flags (`holiday_flag_1 …. holiday_flag_n`), one for each country we need to consider. For instance, October 3rd is a national holiday in Germany, so we set the `holiday_flag` for Germany to `'t'`, while for all other countries we leave it `'f'`. This way, we can run queries like "how did this German holiday affect sales in neighboring countries like Austria and Switzerland?".

4.2.3 Granularity

Granularity is the level of detail of the facts stored in a data warehouse. As mentioned above, if we are only modeling calendar days the time dimension provides a very good approach to track business on a daily basis. But what if we need to add some more precision to the fact table in order to store more temporal details? Can we just increase the granularity of the time dimension to the nearest hour, minute or even second or do we have to give up the ability to navigate by time and to specify seasons, fiscal periods and holidays?

To answer this question let's first take a look at a time dimension that stores all the days in a defined period of time. This dimension will contain a row for each day, which means that a 10-year dimension will contain 3650 rows. Now if we want to track changes to the nearest hour, minute, or second, this could be done in one of the following ways:

4.2.3.1 Increase the Granularity of the Time Dimension

With this approach the time dimension is changed to store all the hours, minutes, or seconds of the specified time period. For a 10-year time dimension this will mean that it will contain approx. 87600 rows (3650 x 24) to store each hour, 5,000,000 rows for each minute, and over 310 million rows for each second.

While this might be an acceptable size for storing hours, this is definitely not the case for minutes and seconds. Moreover, in order to keep the size of the time dimension small and predictable, the duration must be kept constant by deleting old entries when new ones are inserted. This makes the business data stored only semi-periodic (see 4.1.2).

This approach is useful if the granularity is limited to the nearest hour for a not too long period of time (Figure 4.6), which also makes it possible to navigate by hour, for e.g. to see what day times were the best for sales. However, if we need to store the time to the nearest minute or second, I prefer using one of both other approaches.

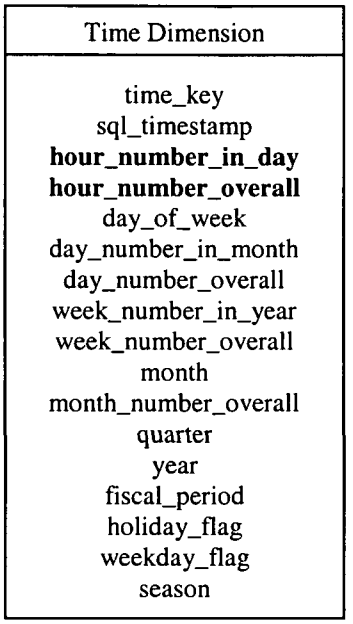


Figure 4.6: A time dimension on hourly basis

4.2.3.2 Add Timestamps to the Fact Table

Adding SQL TIMESTAMPS directly to the fact table provides a very high precision as the granularity of these timestamps can go up to some fractions of the second (Figure 4.7). Occurring events can thus be captured on the second of occurrence, and the start and end of a status are also stored second exact.

time	product_key	customer_key	quantity	price
25-MAR-2003 15:37:13	12265	7657654	5	200
25-MAR-2003 16:15:45	34324	2423555	8	240
03-APR-2003 11:47:02	25254	3545466	3	150

Figure 4.7: A fact table using timestamps

This is good if a very high precision is needed over the navigation features of the multidimensional model. If we choose this approach we have to live with the limitations of SQL TIMESTAMPS and give up the ability to specify seasons, holidays, or fiscal periods to the nearest second.

4.2.3.3 Use Twin Timestamps

This approach combines the advantages of both previous approaches by using two timestamps on each transaction record in the fact table. The first would be an SQL TIMESTAMP as described in the previous paragraph, and the second would be a day id, a foreign key connecting to a calendar day dimension (Figure 4.8).

time	day_key	product_key	customer_key	quantity	price
25-MAR-2003 15:37:13	1910	12265	7657654	5	200
25-MAR-2003 16:15:45	1910	34324	2423555	8	240
03-APR-2003 11:47:02	1919	25254	3545466	3	150

Figure 4.8: A fact table using twin timestamps

The time of day could also be stored in a separate numeric field instead of using a timestamp, or even a separate dimension can be used for the time of day as we will see when we talk about time zones. But anyway it should not be combined into one key with the calendar day as this would make the time dimension simply too large.

This way we can search for very precise time periods, but also navigate to see all transactions that occurred on a holiday.

4.2.4 Daylight Saving Time (DST)

Daylight Saving Time (DST) is the practice of turning the clock ahead as warmer weather approaches and back as it becomes colder again so that people will have one more hour of daylight in the afternoon and evening during the warmer season of the year. DST varies from country to country. Countries in equatorial and tropical climates do not observe DST and the months when the clock is set ahead and back differ between northern and southern hemispheres. We will talk about these variations later when we talk about time zones.

Now we are more concerned about the representation of time on those days when the time is shifted. This happens on two days every year. In the European Union, DST starts the last Sunday in March at 1 am UTC and ends the last Sunday in October at the same time.

These two days must be treated differently in the time dimension because they are different than other days. While all other days of the year have 24 hours and can thus be modeled as shown above the day when the time is set to DST has only 23 hours since the time is set from 0 am directly to 2 am UTC. On the other hand, the day when the time is set back has 25 hours because the hour from 1 am to 2 am is repeated twice (Figure 4.9).

Let's take as an example March 30th 2003 and October 26th 2003. On March 30th there is actually no point in time when it is 1 am. The clock jumps from 00:59:59 directly to 02:00:00. Therefore there is no need to include the 1 am hour in an hourly-based time dimension.

Time is set to DST	Time is set back
00:00	00:00
02:00	1A:00
03:00	1B:00
.	02:00
.	.
.	.
23:00	.
	.
	23:00

Figure 4.9: Hours on DST switch days

On October 26th, however, there are two points in time when it is midnight. The clock goes from 00:59:59 back to 00:00:00 again instead of 01:00:00. We will call this hour 1A. After another hour the time is actually 01:00:00. We will call this hour 1B (Figure 4.9).

For our time dimension shown in Figure 4.6 this means that on all last Sundays in March we only need to insert 23 hours by leaving the 1 am hour, as it does not really exist. The `hour_number_in_day` thus goes from 1 to 23 (Figure 4.10). And on all last Sundays in October we insert 25 hours by repeating the midnight hour. The hour_number_in_day thus goes from 1 to 25 (Figure 4.11). Of course the DST days have to be determined in advance before the time dimension is filled with values.

time_key	sql_timestamp	seconds_in_day	hour_no_in_day	hour_no_overall
45936	30-MAR-2003 00:00:00	0	1	45936
45937	30-MAR-2003 02:00:00	3600	2	45937
45938	30-MAR-2003 03:00:00	7200	3	45938
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
45948	30-MAR-2003 23:00:00	82800	23	45948

Figure 4.10: March, 30th 2003 modeled in the time dimension

time_key	sql_timestamp	seconds_in_day	hour_no_in_day	hour_no_overall
50975	26-OCT-2003 00:00:00	0	1	50975
50976	26-OCT-2003 00:00:00	3600	2	50976
50977	26-OCT-2003 01:00:00	7200	3	50977
50978	26-OCT-2003 02:00:00	10800	4	50978
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
50999	26-OCT-2003 23:00:00	90000	25	50999

Figure 4.11: October, 26th 2003 modeled in the time dimension

The optional attribute `second_in_day` can be useful for the application to correctly determine the timestamp and other time periods on those two special days and it must be interpreted differently on those two days than on normal days. For example, while 7200 seconds would be 2 am on any normal day, it would be 3 am on March 30th 2003, and 1B am on October, 26th 2003. And while the interval between midnight and 2 am is 2 hours on normal days, it would be only 1 hour on March 30th 2003, and 3 hours on October 26th 2003. Depending on the application and the needed queries this attribute can be used or not.

Finally, to make it easier for SQL to determine the days when time is set to DST and back we use another flag `DST_flag` which is zero on normal days, -1 on all last Sundays in March, and +1 on all last Sundays in October. This makes our time dimension look as shown in Figure 4.12.

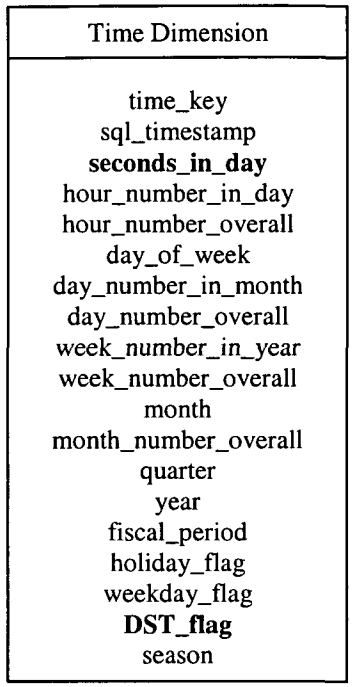


Figure 4.12: A time dimension considering DST

Please note that the described model is using UTC. If you design your time dimension for another time zone, you have to consider the days and times when the time is switched to DST and back.

4.2.5 Time Zones

Different time zones not only mean having different times, but also DST is observed differently in different regions, which makes the design of the time dimension even more complicated.

Here are a few of the world's variations in observing daylight saving time:

- In most parts of North America, clocks are set forward one hour on the first Sunday in April and back on the last Sunday in October.
- The state of Arizona in the U.S. does not observe DST, but the large Navajo Reservation in northeastern Arizona does observe it. The Hopi reservation in the middle of the Navajo reservation, however, does not observe DST.
- In non-equatorial Brazil, DST starts the first Sunday in November and ends the third Sunday in February.
- In the European Union, DST starts the last Sunday in March at 1 am UTC and ends the last Sunday in October at the same time.
- In Russia, the clock is set ahead beginning the last Sunday in March at 2 am local time and set back the last Sunday in October at the same time. Because the clock is already set an hour ahead of standard time, Russians effectively have two more hours of daylight in the summer.
- In Israel and Palestine, DST is observed, but the time of change is decided every year. Israel and the Authority of Palestine sometimes have different start and end dates.

- Jordan has DST time all year.
- Australia's DST starts the last Sunday in October and ends the last Sunday in March. However, Tasmania's DST starts the first Sunday in October along with New Zealand and ends the last Sunday in March. New Zealand ends the third Sunday in March.

As we mentioned at the end of the previous section, the days and times when time is set to DST must be considered in the design of the time dimension. However, business data are not always entered within the same time zone. This is one of the reasons for using a Geography dimension. In a Sales data warehouse this would help to store where a product was sold. But which time should be used: local or universal time?

Now that the web has become an extremely important source of data warehouse data, a source that produces data with the speed of a click, it brings up several issues that are not yet resolved. Data enters the warehouse from thousands of users in different time zones, but must all be stored into the same database.

Generally, it's easier to store the local time than to compute it based on time zones, which is very useful for queries such as "at what time of day were the most orders placed". But as this can be done in any time zone, and as online stores begin to be a very important point of sale the use of universal time might make more sense to compare order times worldwide.

Therefore it is better to store both: local time and universal time. This can be done by adding another copy of the time dimension for universal time (Figure 13).

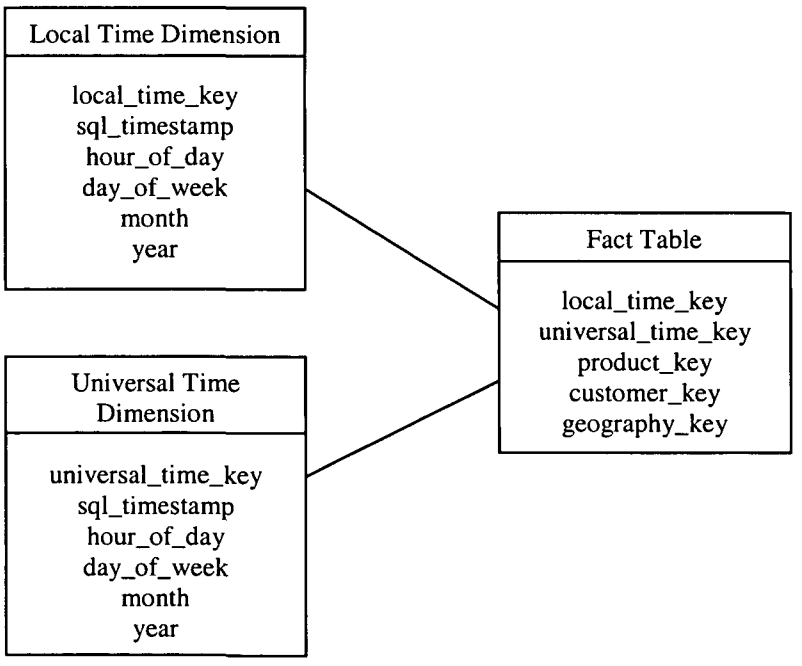


Figure 4.13: A model using two time dimensions for local and universal time

And as mentioned previously, to add more flexibility and granularity the time-of-day can be separated from the day by using two dimensions: a date dimension and a time-of-day dimension for both local and universal time (Figure 14). This gives us altogether four dimensions for representing time. We split the date from the time-of-day because these two components of time have different descriptors. Date relates to calendar and weekdays and seasons, and time-of-day relates to the specific spot we are in within a day. The time-of-date dimension might also be used if we have some specific intervals during the day that we want to assign names to, afternoon, evening, etc.

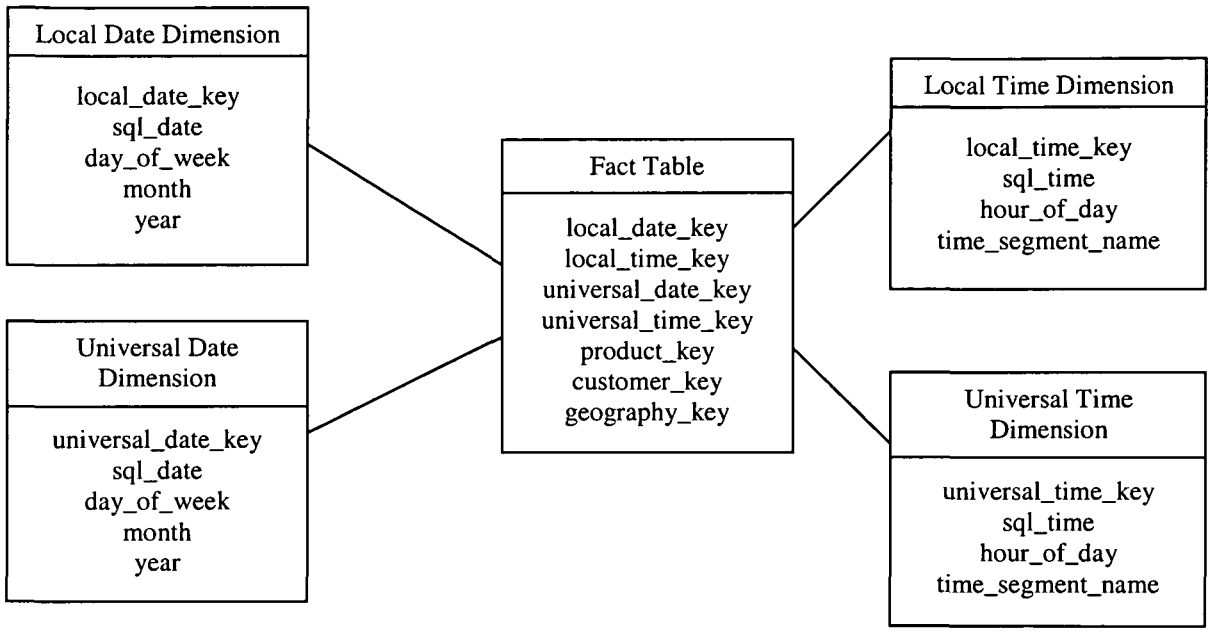


Figure 14: Splitting the time dimension into a date dimension and a time-of-day dimension

This way we can navigate through sales facts by absolute time as well as relative to the customer's time. Having separate dimensions for local and universal time we don't need to implement the time calculation based on time zones into the application logic. This makes our queries more efficient as well.

4.2.6 Examples of Temporal Queries

The following is a list of examples of temporal queries in sales, financial, procurement, and healthcare applications:

Sales examples:

- Show the total amount of sales for product X that occurred in Germany on a weekend or holiday during the second quarter of the current year.
- Compare this amount to the same amount of the same quarter last year.
- Compare this amount to the same amount of the same quarter in France.

Financial examples:

- Did any account have a sequence of deposits followed by next day overdraft withdrawals four or more times within three consecutive months?
- Validate that an overdraft account that is not closed within a week should be in the black within three weeks and stay in the black for one year.

Healthcare examples:

- Has a patient taken a certain series of vaccinations and medications in the right order, W two or more weeks after X, X at most 3 weeks after Y, Z never before W?
- Has this patient waited sufficient time before being given a stronger medication?
- Have tests and examinations been applied to a certain patient in the right order?

Procurement and B2B examples:

- Validate that within a year of the day that sale volume exceeds $200,000 either price drops 15% and sale volume increases 25% or price drops 8% and sale volume will increase by 25% within another year.

Security and Law Enforcement examples:

- Has an individual been arrested more that twice within two consecutive months within a year after release on probation?
- Have three or more first class passengers bought a one way ticket within two weeks after returning from a round trip to the same destination.
- Has every computer user within the network logged off at least once a week?

The following list shows examples of temporal queries for different types of temporal methods, such as temporal projection, temporal slicing, or temporal join:

Temporal Projection examples:

- Show the employment history of employee 'Steve Hanks' (i.e., the departments where he worked and the periods during which he worked there).

Temporal Snapshot examples:

- Show the name and the salary of employees whose salary, on 04-JUN-2004, was above $85,000.

Temporal Slicing examples:

- For all departments, show their manager history in the period starting on 01-FEB-2000 and ending 01-FEB-2001.

Temporal Join examples:

- Count the number of employees each manager managed in the last two years.
- For each department, show the count of their employees on 01-JAN-2003.

Temporal Aggregate examples:

- Show the complete history of the average salary for all the employees in the sales department.

Coalescing examples:

- Find the longest period during which there is no change of title of employees.

Interval examples:

- Find the employees who never got a salary raise from the time they joined to the time their job title was changed.

CHAPTER 5

Time Dimension Updates

As dimensions represent the framework within which factual data is summarized for analysis, changes in analysis requirements or in the structure of the data sources almost always imply changes in the dimensions of the model. These changes are not limited to the addition or deletion of attributes or instances, but they may also involve the hierarchical structure according to which dimensions are organized. All these kinds of dimension updates, together with the data cube maintenance under them, are poorly supported (or not supported at all) in current commercial systems [Hurtado 1999].

Unlike other slowly changing dimensions such as product, customer or geography, changes to the time dimension occur so infrequently that most of the time they are not considered at all in the database design. However, once any small change to the time dimension is needed designers are confronted by the very special nature of this dimension, which makes it very difficult to apply the common techniques for updating other dimensions.

5.1 Techniques to Handle Dimension Updates

Generally speaking, there are three main techniques for handling slowly changing dimensions in a data warehouse [Kimball 1996]: overwriting, creating another dimension record, and creating a current value field. Each technique handles the problem differently. The designer chooses among these techniques depending on the user's needs.

5.1.1 Overwriting

This is a simple and fast technique that is used when the old value of the changed dimension attribute is not interesting. It overwrites the old attribute value in the changed record with the new value. No changes are needed elsewhere in the dimension record and no keys are affected elsewhere in the database. It does not maintain past history and can thus be used to correct wrong values in the dimension tables. For example if incorrect attribute values have been inserted during the initial load or if a holiday flag has been forgotten to be set (see 4.2.2).

5.1.2 Creating another Dimension Record

This is the most common technique used for slowly changing dimensions. It creates an additional dimension record at the time of the change with the new attribute values. This way, history is very accurately maintained and can be recalled very easily because new dimension records automatically partition history in the fact table. The old version of the dimension record points to all history in the fact table prior to that change, while the new version points to all history after that change. There is no need for a timestamp in the dimension table to record that change. In fact, a timestamp in the dimension record may be meaningless because the event of interest is the actual use of the new record in the fact table with the correct new

record key. This technique can track many changes as each change generates a new dimension record that partitions the complete history. The main drawback of the technique, especially for dimensions like product and customer, is the requirement to generalize the dimension key instead of using a unique attribute like the product SKU number or the social security number as a primary key. But as the time dimension uses a separate integer day id, this is not an issue. Also the growth of the dimension table itself is irrelevant for the time dimension, as changes do not occur very frequently.

5.1.3 Creating a Current Value Field

This technique is used occasionally when there is need to track both the old and the new values of a changed attribute. In this case, instead of adding a new dimension record we add a new "current value" field for the affected attribute, so that the old value can be used both before and after the change. This technique is designed to handle only the original and the current values of the changed attribute. Intermediate values are lost. Of course, if there is a need to partition history, then the second technique should be used instead, and all the changes can be tracked. Theoretically, it would be possible to mix both techniques, but this would result in increased application complexity. This technique has very little use for changes to the time dimension.

5.2 Processing During Initial Data Load

Periodically (usually every night) production data load is performed, during which new data from production databases is loaded and integrated into a data warehouse, ensuring the proper representation of prior history. Production databases are usually located physically apart.

When new measurements (facts) are brought into an existing data warehouse, dimensions like product, customer, and time are probably already defined and have a rich history reflecting many "slow changes". The regular ETL (extract-transform-load) processes dimensions first. Any new records from the production source are inserted into the dimension tables and are assigned the next surrogate key in sequence.

Under normal circumstances, no new records are added to the time dimension from the production source unless one or more days have been forgotten during table creation. Otherwise, changes to the time dimension are the result of decisions made by the management, for instance changes to the fiscal periods or adding a new holiday. Usually the time dimension is created for a period of time long enough to cover all expected business transactions.

Existing dimension records that have changed since the last load of the warehouse are detected and the nature of their change examined. An established slowly changing dimension policy will decide, based on which fields have changed, whether the current dimension record should be destructively updated and the old values lost or that a new dimension record possessing the same natural ID should be created using the next surrogate key in sequence.

Finally, it is time for the bulk load of the facts. The natural IDs must be stripped off the fact record and replaced with the correct surrogate keys as fast as possible. Fact records usually use an Oracle timestamp as a natural ID for the day record in the time dimension. Sometimes

small lookup tables are used for each dimension to translate natural IDs to surrogate keys. These tables can be built from the dimension tables in the warehouse RDBMS using statements similar to the ones below:

```
select oracle_date, max(day_id)
from time_dimension
group by oracle_date

select oracle_date, day_key
from time_dimension
where current = 'Y'
```

This way, the lookup tables will contain the surrogate keys that match the new facts and will support rapid lookups.

5.3 Time Dimension Updates

In this section we will discuss the types of possible updates either to the schema (structure) or to an instance of the time dimension, classifying them into two subsets: structural updates and instance updates.

5.3.1 Structural Updates

Structural updates modify the structure of the dimension. We refer to the operators defined in [Hurtado 1999] and show how to apply some of them on the time dimension with specific examples. We start with a simple daily-based time dimension as shown in Figure 5.1.

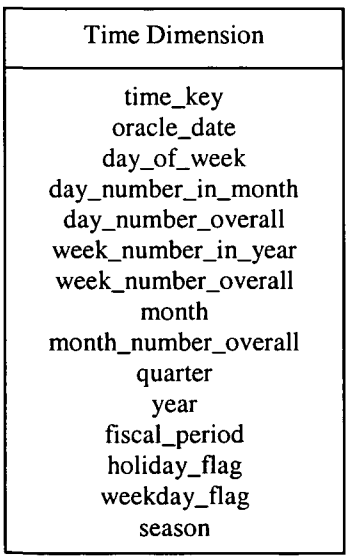


Figure 5.1: The time dimension

This time dimension is connected to the central fact table by a foreign key (Figure 5.2). We will show three examples for structural updates: creation and deletion of a hierarchy level and insertion of a new attribute or flag.

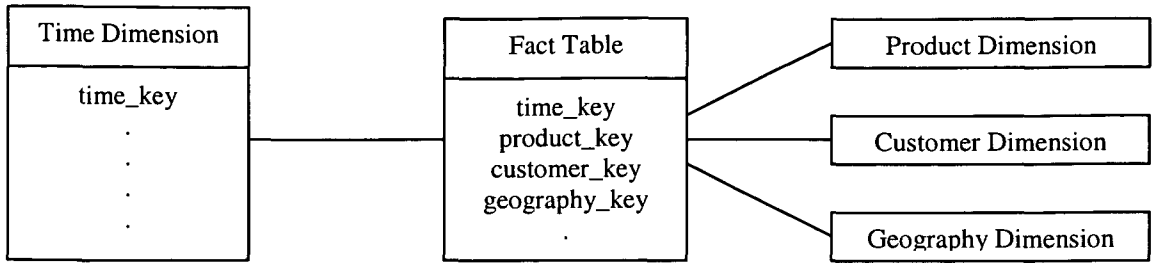


Figure 5.2: A multidimensional model

5.3.1.1 Creating a New Hierarchy Level

In chapter 4, we showed how the granularity of the fact table can be increased by increasing the granularity of the time dimension, for instance from daily to hourly. This requires adding a new level "hour" to the hierarchy schema shown in Figure 5.2. To do this on database level, we need to modify the structure of the time dimension table by adding new fields and constrains (Figure 5.3) before the new hour records can be inserted.

Time Dimension
time_key
oracle_timestamp
hour_number_in_day
hour_number_overall
day_of_week
day_number_in_month
day_number_overall
week_number_in_year
week_number_overall
month
month_number_overall
quarter
year
fiscal_period
holiday_flag
weekday_flag
season

Figure 5.3: A time dimension on hourly basis

First, we modify the table structure by adding two new fields: `hour_number_in_day` and `hour_no_overall`. There is no need to remove the foreign-key constraint in the fact table as the `time_key` of existing day records will remain unchanged and will point later to the midnight hour of each day. New records will be given a new `time_key` while being inserted into the time dimension.

After that we copy the attributes of each existing day record and create 23 new hour records with the same attributes, but with different values for `hour_number_in_day` and `hour_no_overall`. The existing day record will be modified to point to the midnight hour of this day, which makes altogether 24 hours. The following pseudocode can be implemented using any procedural language like PL/SQL to perform the needed change:

```
alter table time_dimension add (
 hour_number_in_day integer not null default 1, -- 1 to 24
 hour_number_overall integer not null; -- first hour is 1);

-- determine the biggest time_key
time_id := select max(time_key) from time_dimension;

-- update the previous day records to become the midnight-hour
   records
update time_dimension
  set hour_number_overall =   row.hour_number_overall*24 -23;

for row in "select * from time_dimension where oracle_timestamp -
sysdate >= 0"

  -- insert 23 more hour records for this day
  for i := 1 to 23
    time_id := time_id + 1;
    row.time_key := time_id;
    row.hour_number_in_day := i + 1;
    row.hour_number_overall := row.hour_number_overall + i;

    -- add i hours
    row.oracle_timestamp := row.oracle_timestamp + i/24;

    insert into time_dimension (time_key, oracle_timestamp,
    hour_number_in_day, hour_number_overall, day_of_week,
    day_number_in_month, day_number_overall, week_number_in_year,
    week_number_overall, month, month_number_overall, quarter, year,
    fiscal_period, holiday_flag, weekday_flag, season)
    values (row.time_key, row.oracle_timestamp,
    row.hour_number_in_day, row.hour_number_overall, row.day_of_week,
    row.day_number_in_month, row.day_number_overall,
    row.week_number_in_year, row.week_number_overall, row.month,
    row.month_number_overall, row.quarter, row.year,
    row.fiscal_period, row.holiday_flag, row.weekday_flag,
    row.season)
  end for
end for
```

An alternative way to increase the granularity of the fact table without having to change the time dimension is to create a separate time-of-day dimension as discussed in 4.2.5, which only stores the hours of the day (Figure 5.4). However, this technique requires a slight change to the fact table by adding a new field for the new hour_key referencing to the time-of-day or hour dimension.

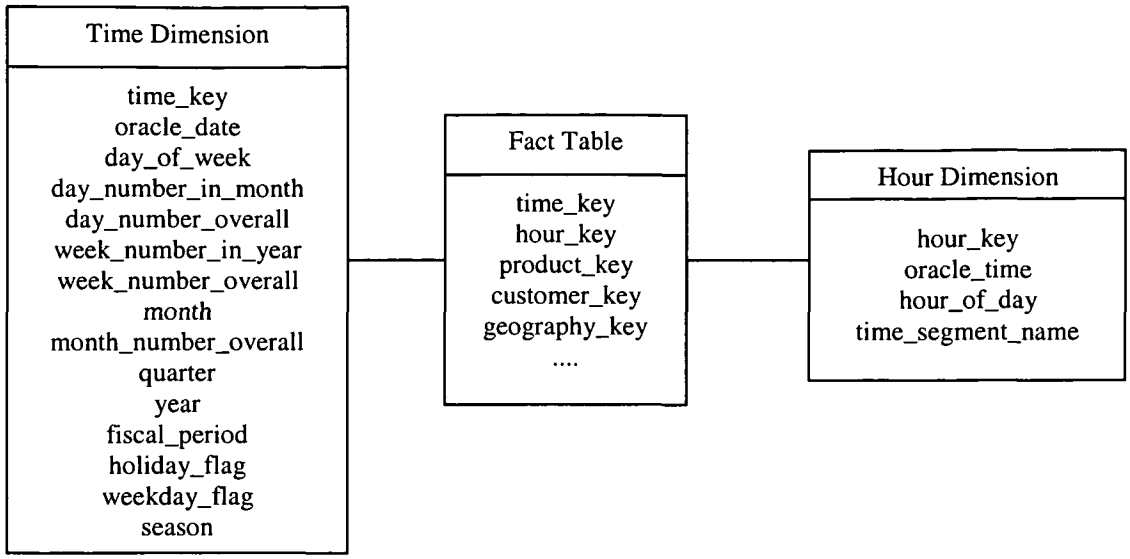


Figure 5.4: Splitting the time dimension into a date dimension and a time-of-day dimension

5.3.1.2 Deleting a Hierarchy Level

Suppose we have an hourly-based time dimension, and our fact table contains for every possible product-store combination a record for each hour. Now suppose that, after a while of running business, the management decides that there is no need to store hourly sales values since sales transactions do not occur that frequently. In this case, it would be enough to store daily values, which would also reduce the size of the fact table and the time dimension and significantly improve query performance.

This requires reducing the granularity of the time dimension to daily basis and modifying the fact table records to represent daily instead of hourly sums. In contrast to the previous example, the focus here is on fact records in the past rather than in the future.

The first thing we do here is to build the daily sums for each product store combination. Here for simplicity, we assume that our cube contains only 3 dimensions: time, product, and geography, and we are concerned about the sales price and quantity. But the idea remains the same with the existence of other dimensions, such as customer or organizational structure. Only the time dimension is handled differently here, but all other dimensions are handled the same way as product and geography.

After building the daily sums, we need to enter them into the fact table. To do this, we update the records for the midnight hour (`hour_number_in_day = 1`) with the daily sum of the same product and the same store. These records become the day-to-day records in the modified fact table.

Now we can delete fact table records no longer needed. In the time dimension we delete all the records for hours later than midnight, leaving only one record for each day. Finally, we delete the hour attributes `hour_number_in_day` and `hour_number_overall` from the time dimension, and our time dimension is now daily based.

This can be done using the following SQL-statements:

```
create view help_view as
  ( select td.day_number_overall, f.product_key,
          sum(f.amount) as total_qauntity
    from fact_table f, time_dimension td
    where f.time_key = td.time_key
    group by td.day_number_overall );

update fact_table f set f.quantity =
  ( select h.total_quantity
         from help_view h, time_dimension td
         where f.time_key = td.time_key and
          f.product_key = h.product_key and
         td.day_number_overall = h.day_number_overall and
          td.hour_number_in_day = 1 );

delete from fact_table where time_key not in
  ( select time_key from time_dimension
    where hour_number_in_day = 1 );

delete from time_dimension
      where hour_number_in_day > 1;

alter table time_dimension remove hour_number_in_day,
hour_number_overall;
```

5.3.1.3 Adding a New Attribute or Flag

Sometimes it is necessary to extend the time dimension with new attributes or flags in order to improve the quality and increase the scope of the reports. Flags for holidays, seasons, fiscal periods or daylight saving time DST introduced in chapter 4 do not always exist from the beginning, but might be added later when the need for them is clear.

Adding a new attribute or flag requires an update to the structure of the time dimension. However, the technique used for doing so is different from those used in the previous examples and depends on the attribute or flag to be added.

We show, as an example, how to add a flag for daylight saving time DST to the time dimension. This DST flag is an integer attribute that specifies, on which day the time is shifted to DST and backwards. The value of this flag is –1 on days when the time is shifted to DST, +1 on days when the time is set back to normal time, and zero on all other days. In the European Union this happens on the last Sunday in March and the last Sunday in October respectively.

The new field `DST_flag` can easily be added to the time dimension using the following statement, which sets the default value to zero for all days.

```
alter table time_dimension add (DST_flag integer not null
         check (DST_flag in (-1,0,1)) default 0);
```

After that, we use a help table to store the days when the time is shifted to DST and backwards. We insert two entries in this table: one for the shift to DST and one for the shift backwards.

```
-- Specifies DST switch that fall on the n-th DAY_OF_WEEK in MONTH.
-- Negative means count backwards from the end.
-- DST_offset is added to the DST_flag of the time dimension

create table DST_switch_days (
   DST_offset      varchar(100) not null check (DST_offset in (-1,1)),
   month           integer not null check (month >= 1 and month <= 12),
   day_of_week     varchar(9) not null,
   nth             integer not null,
   primary key (DST_offset)
);

insert into DST_switch_days (DST_offset, month, day_of_week, nth)
   values (-1, 10, 'Sunday', -1);
insert into DST_switch_days (DST_offset, month, day_of_week, nth)
   values (1, 3, 'Sunday', -1);
```

Then we update the time dimension by setting the `DST_flag` to the value given by the `DST_offset` (-1 or +1) for all the days that fall on the last Sunday of March or October. The following pseudocode can be used to do this and works fine for time dimensions of any granularity, e.g. daily-based or hourly-based.

```
for row in "select DST_offset, month, day_of_week, nth from
DST_switch_days"
    if row.nth > 0 then
    -- If nth is positive, put together a date range constraint
    -- to pick out the right week.
        ending_day_of_month := row.nth * 7
        starting_day_of_month := ending_day_of_month - 6

    update time_dimension
          set DST_flag = row.DST_offset
          where month = row.month
            and day_of_week = row.day_of_week
            and starting_day_of_month <= day_number_in_month
            and day_number_in_month <= ending_day_of_month;
    else
    -- If it is negative, get all the available dates
    -- and get the nth one from the end.
        i := 0;
        for row2 in "select day_number_in_month from time_dimension
                     where month = row.month
                       and day_of_week = row.day_of_week
                     order by day_number_in_month desc"
            i := i - 1;
            if i = row.nth then
                update time_dimension
                  set DST_flag = row.DST_offset
                  where month = row.month
                    and day_number_in_month = row2.day_number_in_month
                break;
            end if
        end for
    end if
end for
```

As we see, structural updates to the time dimension require using different techniques than those commonly used for handling slowly changing dimensions. We also see that in most cases these structural updates imply some instance updates in order be complete. Those

updates differ from case to case and depend on the type of change as shown in the previous examples.

5.3.2 Instance Updates

Instance updates modify one or more instances of a dimension. They add or delete instances to and from a level in a dimension, but also impose some constraints on the way these updates can be performed.

We show some examples for instance updates to the time dimension and how to handle them using the techniques introduced in section 5.1 or basing on them. The examples include setting an existing day to be a holiday, changes in fiscal periods, adding one or more years to the time dimension, and changing the DST switch days.

5.3.2.1 Setting an Existing Day to a Holiday

It doesn't happen every day that a certain day is declared as a new holiday, but if this happens it has to be updated in the time dimension. A recent example for this is the German reunion day on October, 3rd. This day is an official holiday in Germany only since 1991, a year after the Berlin wall has fallen. We go back in time to 1990 and see how this change could have been made to the time dimension.

The time dimension has a boolean attribute `holiday_flag` which is true on holidays, otherwise it is false. This attribute has to be set to `'t'` on all the days that correspond to October, 3rd starting from 1991. This can be done using one of the techniques presented in section 5.1. Here we show the first two techniques: overwriting and creating another dimension record.

Overwriting is most likely the technique that is best to use here. We simply set the `holiday_flag` for October, 3rd to `'t'`, but only for the years starting from 1991. We leave it `'f'` for the years before 1991, since it was not a holiday by this time.

```
update time_dimension
set holiday_flag = 't'
where day_number_in_month = 3 and
      month = 10 and
      year >= 1991;
```

Queries like "show the average sales on holidays for the last 3 years" would include October, 3rd only after 1991, but not before.

Using the second technique we create another dimension record for October, 3rd in all years starting from 1991 with the `holiday_flag = 't'`.

```
-- determine the biggest time_key
time_id := select max(time_key) from time_dimension;

for row in "select * from time_dimension
            where day_number_in_month = 3 and
            month = 10 and
```

```
                year >= 1991"
        time_id := time_id + 1;
        row.time_key := time_id;
        row.holiday_flag := 't';
        insert into time_dimension (time_key, oracle_timestamp,
                hour_number_in_day, hour_number_overall, day_of_week,
                day_number_in_month, day_number_overall,
                week_number_in_year, week_number_overall, month,
                month_number_overall, quarter, year, fiscal_period,
                holiday_flag, weekday_flag, season)
                values (row.time_key, row.oracle_timestamp,
                row.hour_number_in_day, row.hour_number_overall,
                row.day_of_week, row.day_number_in_month,
                row.day_number_overall, row.week_number_in_year,
                row.week_number_overall, row.month,
                row.month_number_overall, row.quarter, row.year,
                row.fiscal_period, row.holiday_flag, row.weekday_flag,
                row.season);
    end for
```

Unlike other dimensions, such as product or customer, creating another dimension record in this case would not bring any advantage since no one would ever ask questions like "how would the business have been if October, 3rd hadn't been a holiday in the last year". And even if someone asks such a question, the answer will not change the fact that this day will remain a holiday in the future, so nothing can be done about that.

5.3.2.2 Changes in Fiscal Periods

The `quarter` and `fiscal_period` fields are text fields containing the organization's designation for what quarter and what fiscal period the particular day falls into. Changes to the fiscal period are decided by the management and affect many day records in the time dimension, usually only for the future. Records representing the past remain unchanged.

However, when reports are generated, you don't always want to see only the current status, exactly as it happened, but sometimes it is necessary to see how it would have been if we hadn't perform this change. For this reason, the previous structure must be maintained, so that it can be recalled easily at any time in the future.

This can be done by using one of the last two techniques: adding another dimension record or creating a current value field. But since many dimension records need to be changed the last technique would be more appropriate because there is no need to add all those new records. Moreover, having both the old and new value in one dimension record makes comparisons easier and faster.

The question that might arise here is whether we should add a last changed field for storing the valid date of this change. Unlike other slowly changing dimensions, where this might be necessary, there is absolutely no need to do this in the time dimension, since this date is actually what the particular dimension record represents. For instance, changing an attribute of the dimension record for April, 1st 2003 becomes valid when this record is first used in the fact table, which is exactly on April, 1st 2003.

5.3.2.3 Adding One or More Years

When the time dimension is created for the first time it is usually populated with a limited number of years. After some time it might be necessary to add some more years to extend the period it covers. This can be done in a way similar to the initial load of the time dimension (See 4.2.2). There is nothing to update here actually, but we will show how to do this using the following pseudocode to add n days starting from January 1st 2008:

```
-- The insertion is driven by the use of the integers table,
-- which just contains a set of integers, from 0 to n.
-- d is the Oracle date of the day we're inserting.

insert into time_dimension
(time_key, oracle_date, day_of_week, day_number_in_month,
 day_number_overall, week_number_in_year, week_number_overall,
 month, month_number_overall, quarter, weekday_flag)
 select n, d, rtrim(to_char(d, 'Day')), to_char(d, 'DD'), n + 1,
       to_char(d, 'WW'), trunc((n + 2) / 7), -- Jan 1, 2008 is a
       Tuesday, so +2 to get the week numbers to line up with the week
       to_char(d, 'MM'), trunc(months_between(d, '2008-01-01') + 1),
       to_char(d, 'Q'), decode(to_char(d, 'D'), '1', 'f', '7', 'f',
       't')
 from (select n, to_date('2008-01-01', 'YYYY-MM-DD') + n as d from
   integers);
```

5.3.2.4 Deleting One or More Years

Deleting some year records from the time dimension usually happens due to storage reasons. As easy as it sounds we would have to handle the existing facts for the years to be deleted. This means, the business data becomes semi-periodic [Devlin 1997]. This is usually an archiving issue, which is out of the scope for this thesis and therefore will not be discussed further.

5.3.2.5 Changing the DST Switch Days

Some countries might decide to change the days when the time is shifted to DST and backwards, as happened in Egypt a few years ago. Instead of May 1st and October 1st, the time is now shifted to DST on the last Thursday in April and backwards on the last Thursday in September, both at midnight.

Of course such a change requires a modification in the time dimension. The `DST_flag` must be updated for the future with the new values. The value of this flag is –1 on days when the time is shifted to DST, +1 on days when the time is set back to normal time, and zero on all other days. First we set it to zero for the days that was previously set to –1 or +1. Here we can use overwriting as the old value will no longer be needed.

```
update time_dimension
set DST_flag = 0
where day_number_in_month = 1 and
      month in (5,10) and
      year >= 1995;
```

After that we go on the same way as we did in section 5.3.1.3 after adding the `DST_flag` to the time dimension. We use the help table `DST_switch_days` and the same pseudocode would work here too.

CHAPTER 6

Modeling Temporal Characteristics with SAP BW

SAP Business Information Warehouse (BW) today is a suitable and viable option for enterprise data warehousing. It is equipped with preconfigured information models and reports as well as automated data extraction and loading methods to provide a common view of enterprise data, which facilitates analysis and interpretation of information. It also enables Online Analytical Processing (OLAP) to format the information of large amounts of operative and historical data.

In chapter 4 we introduced design and modeling techniques for representing temporal information in the data warehouse and solved common problems related to the implementation of the time dimension in business data warehouses, such as representing holidays, seasons and fiscal periods, considering the observation of daylight saving time (DST) and handling different time zones [Hezzah 2004a].

We addressed time dimension updates in chapter 5 and introduced an approach to handle structural and instance updates to the time dimension, and showed how they differ from updates to other slowly changing dimensions [Hezzah 2004b]. We investigated the information model of SAP BW in [Hezzah 2004c] and discussed the role of temporal characteristics as a time reference to business events.

This chapter investigates how the global exchange of time-dependent information can be supported by using SAP BW as an enterprise data warehouse. It provides an overview of the information model of SAP BW with focus on the storage architectural layer. It addresses the time characteristics of SAP BW and introduces a mapping of temporal concepts introduced in previous chapters to SAP BW components. This includes handling different time zones and local time conversion, as well as modeling relevant real-world business issues such as holidays, seasons and daylight saving time (DST). Finally it addresses data archiving issues and investigates the time restrictions on the data archiving function of SAP BW.

6.1 The SAP BW Information Model

Before starting to address the options, tools, and methods available in SAP BW for implementing a solution for modeling temporal information, let's first take a look at the architectural layers of the SAP BW implementation. Figure 6.1 shows the layered architecture of SAP BW accompanied by two administrative architectural components:

- Extraction, loading and transformation (ETL) services layer
- Storage services layer, including services for storing and archiving information

- Analysis and access layer, offering different options for presenting information to end users
- Administration services
- Meta data services

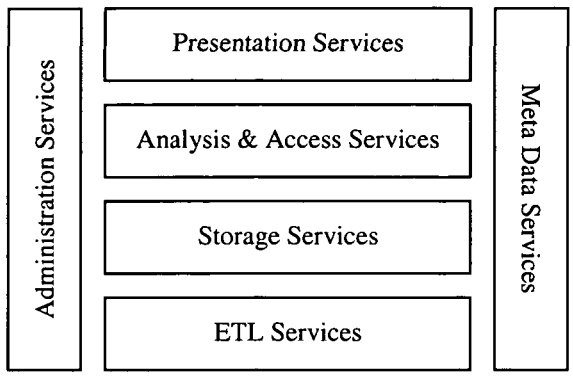


Figure 6.1: SAP BW architecture

SAP BW is built on the basis of a relational OLAP (ROLAP) model. The main structures used for multidimensional analysis in SAP BW are called InfoCubes. The InfoCube Manager generates the InfoCube infrastructure which consists of a fact table and a set of dimension tables, but also the update and retrieval routines according to the definition stored in the meta data repository. It maintains InfoCube data, interfaces with the Aggregate Manager and provides access to InfoCube data for SAP BW reporting and analysis services.

Master data is stored in master data attribute tables, language-dependent text tables, and hierarchy tables. Master data attributes and texts can be defined as time dependent, and hierarchies can be defined as version or time dependent. Generally speaking, master data is data that remains unchanged over a long period of time, e.g. customer, product, etc.

The smallest components in BW are called InfoObjects. They are used to structure the information that is needed to create larger BW objects, such as InfoCubes. There are different types of InfoObjects. These are characteristics, key figures, units, time characteristics, and technical characteristics. Characteristics are sorting keys, such as company code, product, customer group, fiscal year, period, or region. They specify classification options for the data set and are therefore reference objects for the key figures. In the InfoCube, for example, characteristics are stored in dimensions. These dimensions are linked by dimension IDs to the key figures in the fact table.

The characteristics determine the granularity at which the key figures are kept in the InfoCube. The key figures, also known as facts, provide the values that are reported on in a query. Key figures can be quantity, amount or number of items. These values must have units to give them meaning. Time characteristics are characteristics such as date, month, fiscal year, etc.

An operational data store (ODS) object contains key fields (for example, document number/item) and data fields that can also contain character fields (for example, order status, customer) as key figures. The data from an ODS object can be updated with a delta update into InfoCubes and/or other ODS objects or master data tables (attributes or texts) in the same system or across different systems. Unlike multidimensional data storage using InfoCubes,

the data in ODS objects is stored in transparent, flat database tables. Fact tables or dimension tables are not created.

The traditional tool for multidimensional reporting and analysis in SAP BW is the Business Explorer (BEx) Analyzer. The BEx Analyzer is as add-on implemented to Microsoft Excel, and thus combines the power of SAP BW OLAP analysis with all the useful features (e.g. charting) and the Visual Basic for Applications (VBA) development environment of Microsoft Excel. Storing results of queries in Microsoft Excel workbooks, for instance, allows using information in offline mode, sending offline information to other users, or implementing complex VBA applications.

6.2 Time Characteristics in SAP BW

Time characteristics are used within the time dimension of InfoCubes to define the time reference to business events. Since time characteristics are treated internally in a special way in SAP BW, there is no possiblity to create client-specific time characteristics. Table 6.1 shows time characteristics provided by SAP BW.

Table 6.1: Time characteristics in SAP BW

Time Characteristic	Description
0CALDAY	Full date in YYYYMMDD format
0CALMONTH	Month in YYYYMM format
0CALMONTH2	Month in MM format
0CALQUART1	Quarter in Q format
0CALQUARTER	Quarter in YYYYQ format
0CALWEEK	Week in YYYYWW format
0CALYEAR	Year in YYYY format
0FISCPER	Fiscal period including fiscal year variant in YYYYMMM format
0FISCPER3	Fiscal period with fiscal year in YYYYMMM format
0FISCVARNT	Fiscal year variant in VV format
0FISCYEAR	Fiscal year in YYYY format
0HALFYEAR1	Half year quarter in H format
0WEEKDAY1	Day of week in D format

If the InfoCube contains a non-cumulative key figure, then a time-based reference characteristic is needed for the exception aggregation of this key figure.There can be several time characteristics per InfoCube, but only one time reference characteristic. This means, that the time-based reference characteristic is the same for all the non-cumulative key figures of an InfoCube.

The time reference characteristic for an InfoCube, when there are several time characteristics in the InfoCube, is always the "most refined", since all other times in the InfoCube are derived from this. An InfoCube might contain warehouse key figures that should be evaluated for the calendar month and calendar year. In this case, the calendar month is the most refined common time reference characteristic.

It is possible to maintain the time-reference characteristic and the fiscal year variant when updating an InfoCube with non-cumulative key figures. All other time characteristics are

automatically derived from the time-reference characteristic. Therefore, the time-reference characteristic must not be left blank.

There is a difference between complete and incomplete time characteristics: The complete time characteristics are the SAP BW time characteristics calendar day (0CALDAY), calendar week (0CALWEEK), calendar month (0CALMONTH), calendar quarter (0CALQUARTER), calendar year (0CALYEAR), fiscal year (0FISCYEAR) and fiscal period (0FISCPER). They are clearly assigned to a point in time. Only these time characteristics can be used as time reference characteristics, since it must be possible to derive time characteristics automatically from the most detailed time characteristic with the non-cumulative folder.

Incomplete time characteristics, such as 0CALMONTH2, 0CALQUART1, 0HALFYEAR1, 0WEEKDAY1 or 0FISCPER3 can be used in a non-cumulative InfoCube but cannot be a time reference characteristic, since they are not assigned to a specific point in time.

Figure 6.2 gives an overview of the hierarchy of SAP BW time characteristics:

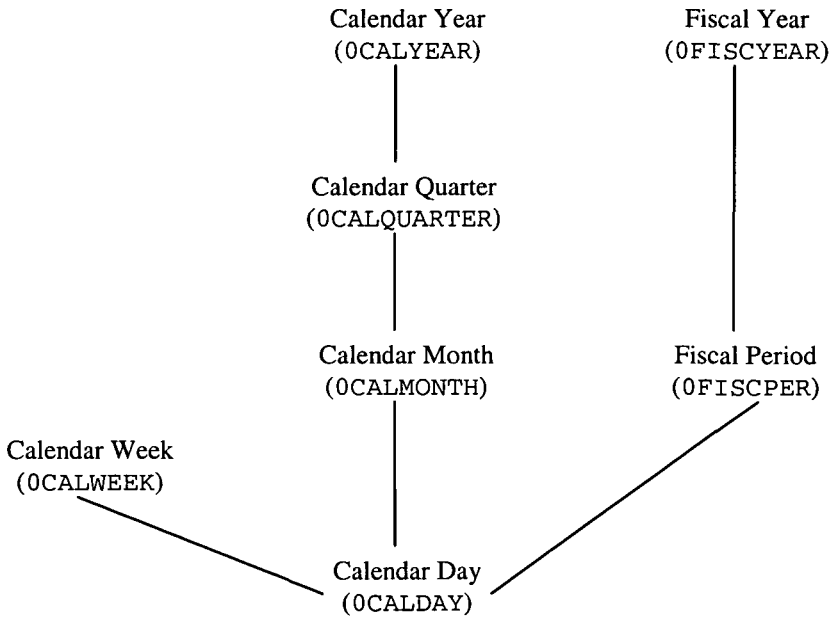


Figure 6.2: Hierarchy of SAP BW time characteristics

If there is a non-cumulative key figure for a week and for a month in the same InfoCube at the same time, the roughest common time characteristic is calendar day. The time characteristic calendar day must be included in the InfoCube, so that it can function as a reference characteristic for time-based aggregation.

6.3 Mapping Temporal Characteristics to SAP BW Components

6.3.1 SAP BW Dimension Structure

Regardless of which class they belong to, InfoCubes consist of key figures and characteristics. The number of dimensions an InfoCube can handle is limited to 16, three of which are used by predefined time, unit and InfoPackage dimensions. Each dimension can hold up to 254

characteristics. To speed up access to the facts, the fact table holds a bitmap index for each dimension ID. It is possible to customize the time dimension by assigning time characteristics.

Although SAP BW provides meta data objects, methods, and tools that allow the implementation of alomst all temporal components of an enterprise data warehouse, the terminology used by SAP for describing InfoCubes has caused some confusion in the data warehouse community. In that community, dimension is typically used for what SAP calls a characteristic, and dimension is used by SAP to refer to a collection of characteristics. This clarifies why a maximum of 13 dimensions in SAP BW is not actually a serious restriction; o single dimension in SAP BW can be composed of up to 254 different characteristics.

Attributes are InfoObjects that exist already, and that are assigned logically to the new characteristic. It is possible to decide for each attribute individually, whether it is time-dependent or not. If only one attribute is time-dependent, a master data table is created. However, there can still be attributes for this characteristic that are not time-dependent. All the time-dependent attributes are in one table, meaning that they all have the same time dependence. All the time-constant attributes are also in one table.

From a technical viewpoint several characteristic values are mapped to an abstract dimension key (DIM ID), to which the values in the fact table refer. The characteristics chosen for an InfoCube are divided up among InfoCube-specific dimensions when creating the InfoCube.

Slowly changing dimensions (e.g. customer or product) are stored in SAP BW master data tables. The master data table can have a time-dependent and a time-independent part. If the dimension contains a characteristic whose value already uniquely determines the values of all other characteristics from a business-orientated viewpoint, then the dimension is named after this characteristic.

The Customer dimension could, for example, be made up of the customer number, the customer group and the levels of the customer hierarchy. The Sales dimension could contain the characteristics 'sales person', 'sales group' and 'sales office'. The time dimension could be given using the characteristics 'day' (in the form `YYYYMMDD`), 'week' (in the form `YYYY.WW`), 'month' (in the form `YYYY.MM`), 'year' (in the form `YYYY`) and 'period' (in the form `YYYY.PPP`).

6.3.2 Time Zones

Local dates and times can only be compared with each other and exchanged if they are in the same time zone. Many global companies, however, work in different time zones, but still need to exchange their data across regional boundaries.

Processes which cover more than one time zone primarily affect logistics functions such as availability checks, production planning, delivery scheduling, statistics and service provision, but they also affect financial accounting in areas such as inter-company transactions, etc.

In chapter 4 we introduced an approach to modeling time zones in the data warehouse, which uses multiple time dimensions for local and universal time. We showed that splitting the time-of-day from the date gives us the capability to navigate sales facts by date and time of both

local and universal time and saves us implementing the time calculation based on time zones into the application logic.

We also extended the time dimension by additional attributes and flags to solve the issue of daylight saving time DST in different time zones. Then in chapter 5 we solved the issue of time dimension updates, and showed how changes to the DST rules can affect the structure as well as single instances of the time dimension.

SAP BW uses a similar approach, which supports the conversion of local dates and times via the time zone function. This function supports using dates and times that are comparable and exchangeable in applications that are implemented worldwide. The only difference is that it integrates DST rules into the time zone configuration and not directly into the time dimension, and the universal time is not additionally stored in a separate dimension, but is calculated based on the time zone via a conversion function (see 6.3.2.1).

All available time zones are maintained in a central table, and are assigned rules for DST observation as shown in Figure 6.3. Rules for time zones, such as the difference from Universal Time Coordinated (UTC), are maintained in a separate table and also assigned to the time zones.

Time zone	Time zone text	TimeZnRule	Difference from UTC	DST rule	Daylight saving rule	Activ
BRZLWE	Brasil West	M0400	- 4 hours	BRAZIL	Brazil	☑
CAT	Central Africa	P0200	+ 2 hours	NONE	NO daylight saving	☑
CET	Central Europe	P0100	+ 1 hour	EUROPE	Europe	☑
CHILE	Chile	M0400	- 4 hours	CHILE	Chile	☑
CHILEE	Chile Easter Island	M0600	- 6 hours	CHILE	Chile	☑
CST	Central Time (Dallas)	M0600	- 6 hours	USA	USA	☑
CSTNO	Central Time No DST	M0600	- 6 hours	NONE	NO daylight saving	☑
CYPRUS	Cyprus	P0200	+ 2 hours	CYPRUS	Cyprus	☑
EET	Eastern Europe	P0200	+ 2 hours	EUROPE	Europe	☑
EGYPT	Egypt	P0200	+ 2 hours	EGYPT	Egypt	☑
EST	Eastern Time (New York)	M0500	- 5 hours	USA	USA	☑
ESTNO	Eastern Time (Indianapo...	M0500	- 5 hours	NONE	NO daylight saving	☑
FLKLND	Falkland Islands	M0400	- 4 hours	CHILE	Chile	☑
GMTUK	Greenwich UK with DST	P0000	+/- 0 = UTC/GMT	EUROPE	Europe	☑

Figure 6.3: Time zones in SAP BW

6.3.2.1 Time Conversion in SAP BW

Generally, users think and act in terms of their local time, and they also expect to use their local time in business transactions. When the SAP BW system is used for global transactions that span time zones, business partners and systems will have different local times. These differences in local times can lead to problems such as late postings and missed batch runs.

For example, a company with its headquarter and database server in Paris requires that all billing documents be posted by 4:00 p.m. Users in the company's office in London might expect that to mean 4:00 p.m. in London, which is 1 hour behind Paris time. Thus any users in London posting billing documents after 3:00 p.m. would be posting their documents too late.

For business processes spanning time zones, inaccuracies of up to 24 hours could occur. To compare the local times of users in different time zones, the SAP BW system represents time differently externally and internally. The external representation of the time is similar to a context-dependent local time. For example, in France, the time is represented in Central European Time (CET) and in Washington in Eastern Standard Time (EST).

Internally, the system normalizes the internal system time to UTC, which serves as a reference time. By normalizing date and time internally, the time zone function eliminates problems that can arise from users working in different local time zones. For some transactions, dates and times are normalized by the system by storing a time zone and a timestamp, which contains the date and time of an event converted from local time to UTC.

Figure 6.4 shows how the Conversion function of SAP BW uses time zone information to transform the local time into universal time. Here, the requested delivery date of `3 Apr 2004 13:00:00 CET` for a ship-to address in Germany receives the timestamp of `3 Apr 2004 12:00:00 UTC`.

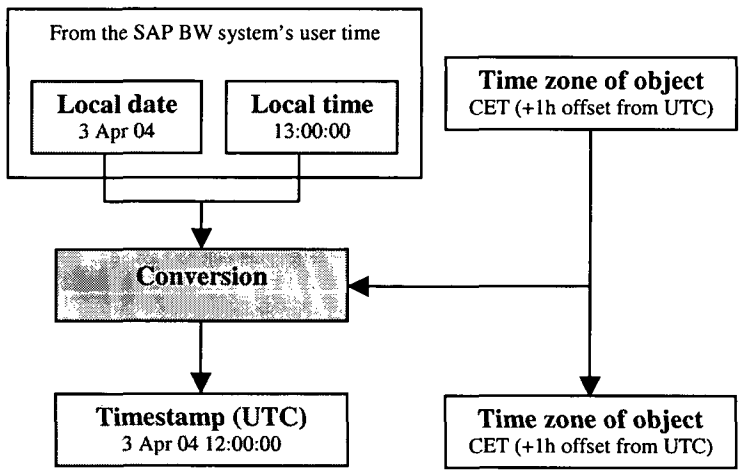


Figure 6.4: Time conversion function of SAP BW

To determine the time zone of an object in SAP BW, the system uses a series of decision rules. By determining an object's time zone, the system can display a timestamp of the object in any local time. To ensure consistent determination of time zones and efficient performance, this process is performed by a central function depending on the location of the object.

SAP BW uses a 24-hour clock with the local time and local date of the object (here the ship-to address) from the user interface with the object's time zone in oder to calculate the timestamp. To display the object's local time and date, the SAP BW system uses the object's time zone, which is stored with the timestamp, and goes through the process backwards. For application programs, it is generally sufficient to us a timestamp accurate to the second.

The timestamp's external representation is similar to the date and time representation. The user is provided the same options for displaying the timestamp as for the date and time:

- `DD.MM.YYYY hh:mm:ss (03.04.2004 14:36:25)`
- `MM/DD/YYYY hh:mm:ss (04/03/2004 14:36:25)`
- `MM-DD-YYYY hh:mm:ss (04-03-2004 14:36:25)`

- YYYY.MM-DD hh:mm:ss (2004.04-03 14:36:25)
- YYYY/MM/DD hh:mm:ss (2004/04/03 14:36:25)

The total output length is 19 characters. The system supports displaying times without seconds, but it does not support displaying times as 'a.m.' or 'p.m.'.

Internally, the data types for date and time are combined by the system to create the 14-character timestamp (8 characters for the date and 6 for the time). Combining date and time allows the system to sort timestamps correctly by the use of date (year-month-day) or time (hour-minute-second). The range of values allowed for the timestamp is '01.01.0001 00:00:00' to '31.12.9999 23:59:59'. The system always uses a 24-hour clock to avoid confusion with a.m. and p.m. time designations, and the system's initial value for the timestamp, which corresponds to midnight, is zero or 00:00:00 instead of 24:00:00.

The following example describes dates and times on inter-company documents between two companies located in different time zones. The local date is different for the two companies:

Company code A: 2000, Location: Los Angeles, Date: 15.04.04, Time: 19:36:03
Company code B: 5200, Location: Melbourne, Date: 16.04.04, Time: 10:36:03
System Date: 15.04.04

The document is associated with a single day, therefore, the document date, posting date and entry date have the same value for both companies, although they may differ from each other. The determination of the posting date depends on the type of transaction. The system records this transaction for both companies with the date of the system at the time of issue. Once the document has been received in the receiving company, the time zone function will propose a posting date based on the local date and time zone of the user who entered the document.

However, considering dates alone is not sufficient to ensure exact time calculations. For time-critical processes, dates with times replace dates without times. A date standing alone, could easily result in a one day inaccuracy (for example, depending on the time of day, 3 April in Melbourne may still be 2 April in Los Angeles). For a date without a time, an inaccuracy related to time zones can be as long as 48 hours in some extreme cases. For time calculations, an accurate duration (for example, hours and minutes instead of days) must be used. Otherwise, chain calculations could be inaccurate by several days.

6.3.2.2 Daylight Saving Time in SAP BW

Some time zones observe daylight saving time (DST) and use a "DST rule" for calculation purposes. For these time zones, clocks are normally set forward one hour to make better use of the longer daylight hours in the late spring, summer and early fall.

SAP BW uses a structure for DST observation, which is slightly different from the one we introduced in chapter 4 and chapter 5. This structure integrates the rules for DST into the time zone rules, which makes maintenance and updating easier, but otherwise doesn't have any comparative functional advantage over the approach we previously presented. However, for global companies using data in different time zones, the calculation of DST offsets is this way integrated into the application logic and doesn't need to be considered on database level.

SAP BW introduces rules to maintain DST start and end dates as well as the time shifts caused by DST. These rules result in the following structure for a time zone in the system (Figure 6.5):

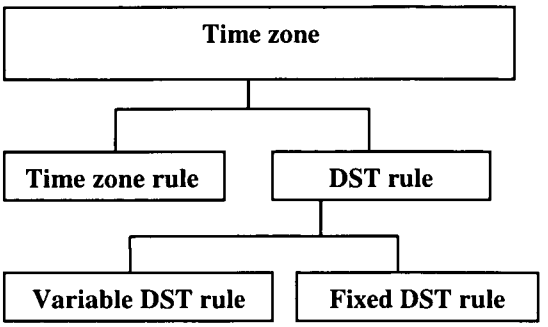


Figure 6.5: Structure for DST rules

DST rules (Figure 6.6) define the offset of DST relative to the time zone's standard time (for Europe and USA +1 hour). It does not define the start and end dates of DST. These rules are then assigned to the different time zones as already shown in Figure 6.3.

DST rule	Diff. DST	Daylight saving rule	Active
EGYPT	01:00:00	Egypt	☑
EUROPE	01:00:00	Europe	☑
IRAN	01:00:00	Iran	☑
IRAQ	01:00:00	Iraq	☑
ISRAEL	01:00:00	Israel	☑
JORDAN	01:00:00	Jordan	☑
LBANON	01:00:00	Lebanon	☑
NEWZEA	01:00:00	New Zealand	☑
NONE	00:00:00	NO daylight saving	☑
PARAGU	01:00:00	Paraguay	☑
SYRIA	01:00:00	Syria	☑
UK	01:00:00	UK	☑
USA	01:00:00	USA	☑
VERM02	00:30:00	ST = WT+30 minutes	☑

Figure 6.6: DST rules

Variable DST rules (Figure 6.7) define how the system calculates the start and end dates of DST. These rules can always be changed, so there is no need to maintain DST start and end dates for every year. For cases in which DST is not defined by variable rules, fixed DST rules define the start and end dates for a specific year.

DST rule	Valid from	Start mon.	Start day	StrtDayMo.	Strt. time	End month	End day	EndDay/Mo.	End time
CHILE	1998	10	1	2	02:00:00	3	1	2	02:00:00
CYPRUS	1998	3	1	5	02:00:00	9	1	5	02:00:00
EGYPT	1998	4	6	5	02:00:00	9	6	5	02:00:00
EUROPE	1990	3	1	5	02:00:00	9	1	5	03:00:00
EUROPE	1996	3	1	5	02:00:00	10	1	5	03:00:00
ISRAEL	1998	3	6	3	02:00:00	9	6	3	02:00:00
LBANON	1999	3	1	5	02:00:00	9	1	5	03:00:00
NEWZEA	1998	10	1	1	02:00:00	3	1	3	02:00:00
UK	1990	4	1	1	02:00:00	10	1	1	03:00:00
USA	1990	4	1	1	02:00:00	10	5	1	03:00:00
VERM10	2222	10	1	1	02:00:00	3	1	5	03:00:00
VERP10	2222	3	1	1	02:00:00	10	1	5	03:00:00

Figure 6.7: Variable DST rules

Rather than distinguishing between two separate time zones (one for winter and one for summer), only one time zone indicator is used in SAP BW which includes the DST rule when applicable. The geographical assignment of DST rules and time zone rules can be performed at country, region, or even postal code level.

The switch backwards from DST to "winter time" can cause problems because the clocks are set back by one hour, which means that an hour is repeated (see 4.2.4). For applications that use timestamps, this can cause the following problems:

- Timestamps from different real times can have the same value
- The timestamps do not necessarily reflect the sequence in which system events really occurred

In chapter 4 we solved this problem by introducing the 23-hour and 25-hour day, which simply deletes one hour from all days on which time is switched to DST, and inserts an additional hour on all days on which the time is switched backwards. Applications that use timestamps based on UTC are not affected by this problem.

Since not all timestamps in the SAP BW system are based on UTC, SAP has until recently recommended shutting down the system during this time to avoid the problem described above. The new solution to this problem is the DST Safe Kernel, which makes time during the "repeated hour" run at half the usual speed. This means that the system can rely on timestamps in the correct sequence without duplicates, even if it is not using UTC, which solves many issues related to the system availability of SAP applications.

6.3.3 Holidays in SAP BW

In chapter 4 we introduced among other things a practical approach, which models relevant real-world business issues, such as holidays, seasons, and fiscal periods, by extending the time dimension with new attributes and flags. In order to consider holidays in different countries or in different time zones we used multiple holiday flags (`holiday_flag_1 …. holiday_flag_n`), one for each country we needed to consider. Integrating these attributes into the time dimension effectively reduces query execution time and provides more

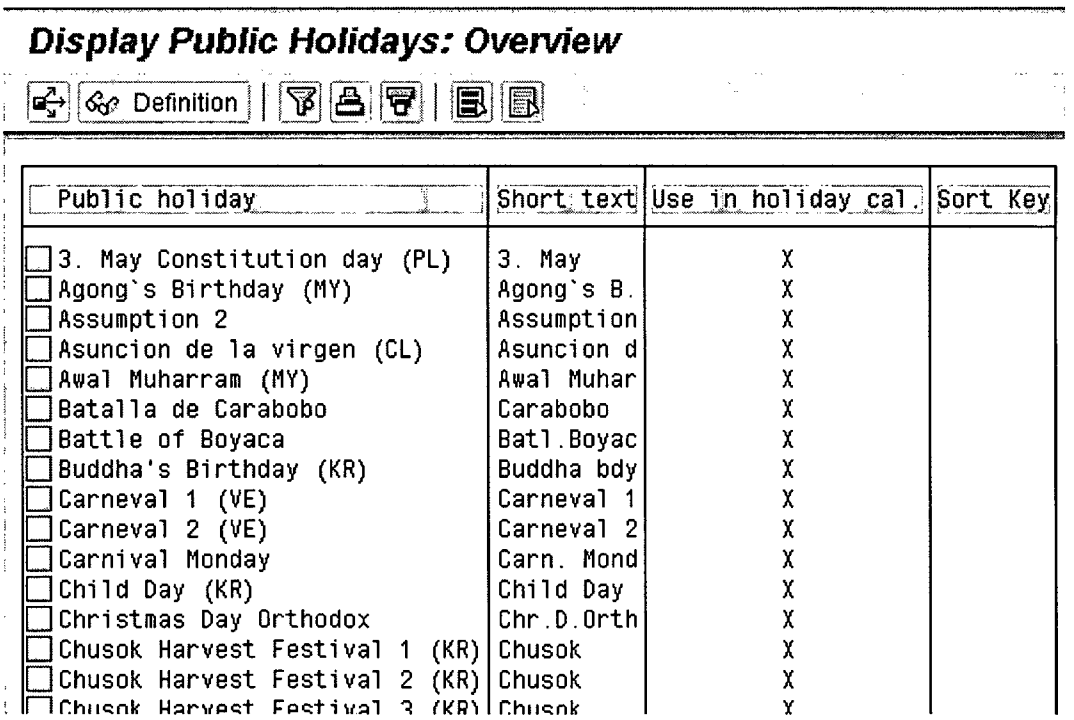
functionality than using conventional RDBMS tables, for instance, navigating data by holidays and non-holidays. Here we investigate how this issue is handled in the current implementation of SAP BW.

While SAP BW integrates seasons and fiscal periods into the time dimension as shown in 6.2, it still stores holiday data in a separate table called public holidays. This table is used by two other tables, public holiday calendar and factory calendar, to define holiday rules.

The public holiday and factory calendar is a central module in the SAP BW system. It is used in many areas, such as logistics and human resources. The calendar system consists of the following components:

Public holidays: Contains the definition of public holidays, calculations rules for date, religious denominations, etc. (Figure 6.8). It consists of the following attributes:

- Public holiday type
- Date or calculation rule
- Public holiday text (short or long)
- If required: Sort criterion, religious denomination or public holiday class



Public holiday	Short text	Use in holiday cal.	Sort Key
3. May Constitution day (PL)	3. May	X	
Agong's Birthday (MY)	Agong's B.	X	
Assumption 2	Assumption	X	
Asuncion de la virgen (CL)	Asuncion d	X	
Awal Muharram (MY)	Awal Muhar	X	
Batalla de Carabobo	Carabobo	X	
Battle of Boyaca	Batl.Boyac	X	
Buddha's Birthday (KR)	Buddha bdy	X	
Carneval 1 (VE)	Carneval 1	X	
Carneval 2 (VE)	Carneval 2	X	
Carnival Monday	Carn. Mond	X	
Child Day (KR)	Child Day	X	
Christmas Day Orthodox	Chr.D.Orth	X	
Chusok Harvest Festival 1 (KR)	Chusok	X	
Chusok Harvest Festival 2 (KR)	Chusok	X	
Chusok Harvest Festival 3 (KR)	Chusok	X	

Figure 6.8: Public holidays

If other public holidays are needed, it is possible to add them by maintaining the public holiday definition and copying them to new or existing public holiday rules.

Public holiday calendar: Contains any composition of public holiday rules (Figure 6.9). Here it is possible to assign any public holiday required to a public holiday rule, which has the following attributes:

- Calendar ID
- Calendar description
- Period of validity (From year, To year)

Display Public Holiday Calendar: Overview

Definition | Calendar

ID	Holiday calendar	Valid from	Valid to	Use in holiday cal.
01	Public holiday calendar Schleswig-Holstein	1990	2009	
02	Public holiday calendar Hamburg	1995	2010	
03	Public holiday calendar Lower Saxony	1995	2010	
04	Public holiday calendar Bremen	1995	2010	
05	Public holiday calendar North Rhine-Westphalia	1995	2010	
06	Public holiday calendar Hesse	1995	2010	
07	Public holiday calendar Rhineland-Palatinate	1995	2010	
08	Public holiday calendar Baden-Württemberg	1980	2010	X
09	Public holiday calendar Bavaria	1995	2010	
10	Public holiday calendar Saarland	1995	2010	
11	Public holiday calendar Berlin	1995	2010	
12	Public holiday calendar Brandenburg	1995	2010	

Figure 6.9: Public holiday calendar

Display Factory Calendar: Overview

Definition | Calendar

ID	Calendar ID	Valid from	Valid to
01	Factory calendar Germany standard	1990	2009
99	International factory calendar	1997	2010
AJ	Annual arrangement	1996	2011
AK	Annual arrangement at the start of the year	1996	2010
AM	Monthly settlement	1996	2011
AN	Monthly arrangement at the start of the month	1996	2010
AO	Arrangement Mondays and Fridays	1996	2010
AR	Factory Calendar - Argentina standard	1996	2001
AT	Factory calendar Austria standard	1996	2010
AU	Factory calendar Australia standard	1996	2010
BE	Factory calendar Belgium standard	1996	2010
BR	Factory calendar Brazil	1996	2001
CA	Factory calendar Canada standard	1996	2010
CH	Factory calendar Switzerland standard	1996	2010
CN	Factory calendar China	1996	2010
CO	Factory calendar Columbia	1997	2010
CZ	Factory calendar Czech Republic standard	1996	2010

Figure 6.10: Factory calendar

Factory calendar: Contains a definition of workdays including special regulations, under the assignment of a particular public holiday calendar (Figure 6.10). The following attributes are maintained:

- Factory calendar ID
- Factory calendar description

- Period of validity (From year, To year)
- Start no. factory date (Number from which the factory date is incremented for each workday, the default value is "0")

The main drawback of this structure is that it doesn't support navigating data efficiently by holidays or non-holidays. Also separating the holiday definition from the time dimension increases query execution time and decreases overall performance. Besides, it will be too complex if we want to look at data, not just on holidays, but also on different seasons, fiscal periods or weekdays. These attributes are stored in different tables and must be joined with the fact table.

However, using the public holidays and factory calendar automatically eliminates irrelevant holidays since only holidays assigned to a holiday rule are considered in the executed query. This way, not all entries in the public holidays table need to be examined by the query. Moreover, for global organizations, which are the main target group of SAP BW, it is a big advantage being able to store all holidays of all countries and regions in a single database table and assign holiday rules to time zones to include only a subset in any query.

6.3.4 Data Archiving

6.3.4.1 Features of the Archiving Function

Data Archiving enables SAP BW users to simplify InfoCube and ODS object administration and improve performance. For this purpose, SAP BW provides the Archive Development Kit (ADK), a tool that develops archiving solutions and prepares the runtime environment for archiving. Its main function is to read and write data to and from archive files. The ADK guarantees that the archived data is both release- and platform independent.

Data archiving is used in SAP BW for data that is no longer needed in running analysis processes, but is still important or may need to be analyzed once more in exceptional circumstances. It allows the user to archive data from both InfoCubes and ODS objects.

The function Archiving, which is available in InfoCube and ODS object maintenance, allows maintaining the properties of an archiving object. An archiving object is a logical object containing related business data in a database that can be read from the database with a write program, and, after successfully being archived, can be deleted from the database using a corresponding delete program. It is possible to select both the selection characteristics and the maximum size for an archive file. An archiving object is generated from these properties when a data target is archived.

It is also possible to schedule archiving sessions and call up further archiving functions. An archiving session consists of a write-, delete-, and a storage phase. Archive files are stored during the write phase for a particular archive object, deleted from the database after the verification phase, and then stored in a storage system according to its definition in Customizing.

Special archiving features include:

- The selection criteria control the archiving. No Implementation Guide (IMG) entries (for example, document age) are required, as in data archiving.
- Data for InfoCubes is not stored in multidimensional tables, but in flat structure.
- For ODS objects only a table content is stored and not the change log content.
- Archiving information is not transferred to the data targets with data that has already been updated. That is, data from data targets stored nearby remains where it is and is not archived.
- The delete phase is based in the selected archiving session selection criteria. Here, existing aggregates are either adjusted or newly created. The delete phase can take place at a different time to the write phase.

6.3.4.2 Time Restrictions for Archiving

Using relative and absolute time restrictions reduces the maintenance workload for data archiving. Variants with relative time restrictions are designed for periodic scheduling of archiving runs, for example, through a periodic event.

Using relative time restrictions, it is possible to specify a selection condition (for the selected time characteristic) that is relative for the data for the execution of the write program:

- *Only Records Older Than:* Here it is possible to determine the upper limit of the time slot by specifying a time period relative to the current date. The time unit for this time period can be specified in days, weeks, months, quarters, or years.

- *Only Complete:* The upper limit can be rounded down here by selecting another, additional time unit for the end of a time period. In doing so, it is also possible to archive, for example, only complete months or years when selecting a date field as a selection characteristic for the time slot. The time slot that is defined this way initially contains the complete time period before the upper limit that was calculated in this way. Optionally (when protecting the archived data areas in an ODS object, this option is fixed and set), time slots that were already archived can appear from the selection. In this case, the lower limit of the time slot normally comes from the lower limit of the time slot for the preceding archiving run.

Using absolute time restrictions, it is possible to specify additional selection conditions for the time characteristic. For the actual selection of the archiving run, the intersection of the relative and the absolute time restrictions is then taken. In doing so, absolute time restrictions can be used especially with exclusive selection conditions to exclude specific time periods from the selection. If the time period for the relative time restrictions is initially left (=all values), then only the absolute time restrictions are valid.

The following example shows how to specify a relative time restriction. Suppose we have made the following specification in the year 2004:

```
Only records older than:          2 years
Only complete:                    Fiscal years
```

This selection hast the result that only data is archived from complete fiscal years that is older than two years. From the current time this would be all records that were loaded prior to 2002.

With absolute time restriction, the specified time restriction is independent of the current time. For example, we have made the following specification:

```
Fiscal year:                    2000 to 2002
```

This selection result in data being archived that was loaded between 2000 and 2002, independently of the current date.

If we have specified both relative and absolute time restrictions, the intersection of both is calculated. Using the specifications made in the two previous examples for both relative and absolute time restrictions, the intersection means that only data from 2000 and 2001 is archived.

In order to be suitable for supplying SAP BW with historical data, every customer-defined information structure can be converted to an InfoSource for SAP BW and its data transferred to the appropriate BW InfoCubes using a one-time update.

If customers are using atomic structures that are further aggregated daily using suitable loading programs, the deltas that are deleted periodically from the atomic structures after aggregating must be suitably archived. This archive can then be imported to an information structure or InfoSource and transferred to BW.

If the data does not fit into an existing information structure, or there is no suitable archive data for the document data that was pre-aggregated during atomic updating, previously-comulated historical information can only be made available in BW by statistical setup to an information structure/InfoSource created for that purpose.

CHAPTER 7

Data Warehouse Process and Workflow Management

This chapter discusses the role of temporal aspects in data warehouse process and workflow management and the impact of data warehousing on supporting business intelligence to meet the expectations of the enterprise.

7.1 Data Warehouse Process Management

"The design and implementation of operational data warehouse process is a labor-intensive and lengthy procedure, covering thirty to eighty percent of effort and expenses of the overall data warehouse construction" [Shilakes 1998], [Demarest 1997]. In order to efficiently support the design and implementation tasks by the meta model, it is important to mention two aspects of data warehouse processes: *complexity of structure* and *relationship with the involved data.* In the proposal presented in [Vassilidis 2001], "the logical perspective is capable of modeling the structure of complex activities and capture all the entities of the widely accepted Workflow Management Coalition Standard [WfMC 1998]. The relationship of data warehouse activities with their underlying data stores is taken care of in terms of SQL definitions."

This idea somehow reverts the classical belief that data warehouses are only a collection of materialized views. In previous research, it has been common to assign a simple view definition directly to a data warehouse table. Although this abstraction is elegant and sufficient for examining other strategies for view maintenance, it is unable to capture real world processes in a data warehouse environment. However, "it is possible to deduce the definition of a table in the data warehouse as the outcome of the combination of the processes that populate it" [Vassilidis 2001]. This new definition complements existing approaches, as it provides the operational semantics for a data warehouse table and its contents, while the existing definitions give an abstraction of its intentional semantics.

The conceptual process perspective traces the reasons behind the structure of the data warehouse. The demand-oriented concept of dependencies as in the Actor-Dependency model introduced by [Yu 1994] is extended with the supply-oriented notion of suitability that fits well with the redundancy often found in data warehouses. It provides an extension to the Actor-Dependency model that has generalized the notion of role to uniformly trace any person, program or data store participating in the system.

The implementation of the metamodel in an object logic facilitates exploiting the query facilities of the repository in order to provide the needed support for checking the consistency of the data warehouse design. The deductive capabilities of ConceptBase [Jarke 1995] provide the facilities to avoid assigning all the interdependencies of activity roles in the conceptual

perspective manually. It would be sufficient to impose rules to deduce these inter-dependencies from the structure of activities and data stores.

Although the design and implementation of the data warehouse are mainly done in a structured environment, the administration of the warehouse usually deals with problems that evolve in a rather ad-hoc fashion. For instance, during the loading of the warehouse contingency treatment is required for the efficient administration of failures. In such events, not only the knowledge of the structure of a process is important; but also the specific traces of executed processes must be tracked down. In case of failure, not only the causes of the failure, but also the progress of the loading process by the time of failure has to be detected, in order to efficiently resume the operation. However, failures during the loading of the warehouse might have critical consequences as far as problems in the warehouse environment are concerned. This brings up the issue of data warehouse quality and the ability of a metadata repository to trace it in an expressive and usable way.

7.1.1 Traces and Complexity

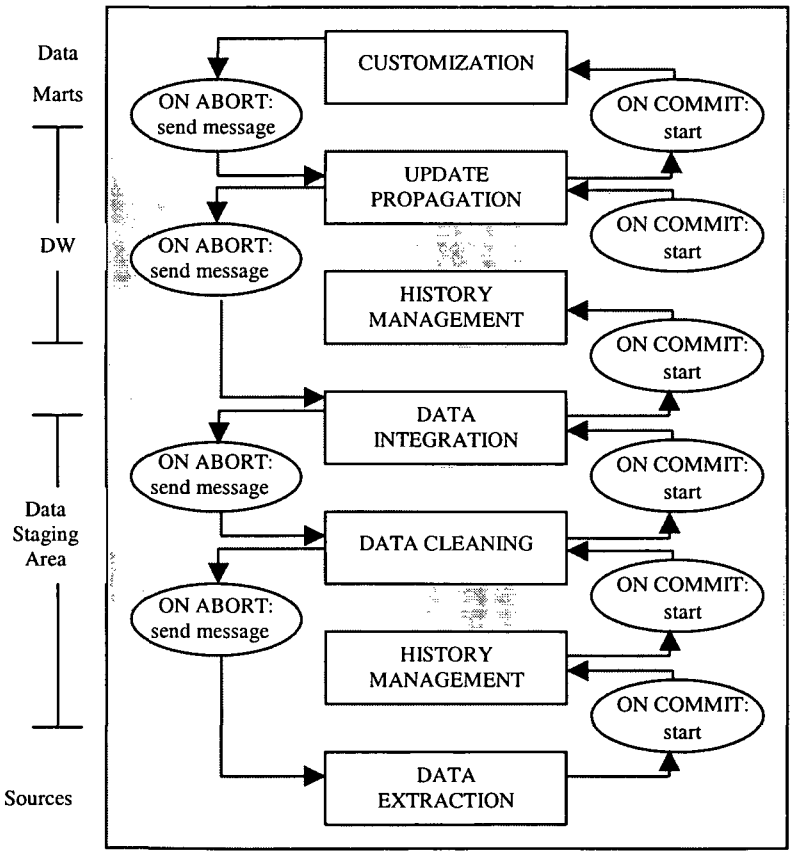


Figure 7.1: The data warehouse refreshment process

Operational data warehouse processes are complex in terms of tasks executed within a single process, execution coherence, contingency treatment, etc. A process metamodel should be able to capture these kinds of complexity. In Figure 7.1 [Vassilidis 2001] the refreshment process of the data warehouse is shown, as described in [Bouzeghoub 1999]. The refreshment process is composed of activities, such as Data Extraction, History Management, Data

Cleaning, Data Integration, History Management, Update Propagation and Customization. Each activity could be executed on a different site. The activities are connected through rules, represented by arrows. The dark background in Figure 7.1 shows a composition hierarchy in the set of data warehouse operational processes. It is common to isolate only a small subset of the overall processes of the warehouse. Any meta model must be able to support zooming in and out the process structure, in order to achieve this functionality.

The main reason for these inspections is to avoid or recover erroneous execution during runtime. Not only the process structure is important; but also the specific traces of executed processes should be tracked down. If the repository could capture this kind of information, it gets more value because: 1. the data warehouse stakeholders can use it for design purposes (e.g., to select the data warehouse objects necessary for the performance of tasks) and 2. decision makers can relate the data warehouse objects to decisions, tools and the facts which have happened in the real world.

7.1.2 Data-oriented Operational Data Warehouse Processes

Data warehouse activities are usually data intensive when it comes to pushing data from the sources to the data warehouse tables or the client data marts. This can be justified by looking at the most common operational processes:

- data extraction: used for extracting the information from the legacy systems;
- data transfer and loading: used for the instantiation of higher levels of aggregation in the warehouse with data from the sources or lower levels of aggregation;
- data transformation: used for transforming the propagated data to the required format;
- data cleansing: used to ensure the data consistency in the data warehouse (i.e., that the data respect the database constraint and the business rules);
- computation: used for derivation of new information from the stored data (e.g., aggregation, querying , business logic, etc.).

To handle the complexity of loading process in the data warehouse, specialized Extraction-Transformation-Loading (ETL) tools can be found today. They mainly focus on:

- identifying relevant information at the source side,
- extracting this information,
- customizing and integrating the information originating from multiple sources into a common format,
- cleaning the resulting data set based on database and business rules, and
- propagating the data to the data warehouse and the data marts.

According to [Shilakes 1998], "ETL and data cleaning tools cover a labor-intensive and complex part of the data warehouse processes, estimated to cost at least one third of effort and expenses in the budget of the data warehouse". [Demarest 1997] mentions that "this number can rise up to 80% of the development time in a data warehouse project". However, due to the complexity of these tools, many organizations prefer using own developments to perform ETL and data cleaning tasks.

7.2 Data Warehouse and Workflow Management

7.2.1 Workflow Management Systems

Workflow management refers to managerial activities performed manually or automatically such as modeling the work processes, monitoring work processes, controlling the routing of tasks, allocating tasks to roles assumed by employees, and alerting employees and managers of exceptional events in the system.

Workflow Management Systems (WfMSs) are software platforms that allow the definition, execution, monitoring, and management of business processes. WfMSs log all events that occur during process execution. Therefore, workflow logs include a significant amount of information which can be used for analysing process executions, understanding the causes of low- and high-quality process executions, and rating the performance of business partners and internal resources.

Many organizations are increasingly using WfMSs for improving the efficiency of their processes and reducing costs. WfMSs are able to log events which occur during process executions, for instance the start and completion time of each activity, its input and output data, and the resource that executed the activity. This kind of analysis allows business and IT managers to identify problems and inefficiencies, and to find solutions. But most WfMSs provide only basic log analysis functionality, such as retrieving the number of process instances completed in a given period of time and their average execution time. For the users to get more comprehensive reports, they usually have to configure their reporting tools and write some queries on the WfMSs logs to retrieve the desired data.

This approach, although providing basic reporting functionality, it usually requires a considerable configuration effort to write the required queries and to extract the desired information. Besides, WfMSs logs are often not designed for OLAP applications, might contain incorrect information that must be checked and cleansed, and do not support aggregating data from multiple data sources.

To overcome these limitations, [Bonifati 2001] designed and implemented a warehouse of workflow execution data and called it Workflow Data Warehouse (WDW). The goal of this WDW is to develop an optimized solution for HP Process Manager (HPPM), which is also applicable to any other WfMS. The WDW must be easy to install and to use, and has to perform adequately under different conditions, (e.g., different log sizes, or different data loading and aggregation requirements). However, warehousing workflow data still presents several challenges:

- Multiple related fact types: Workflow executions could generate different types of facts about workflow activities, instances, and resources. These facts are related to each other. For instance, activities are executed in the context of a specific workflow instance. The presence of multiple, related types of facts affects both the design of the warehouse schema and the data loading process, due to the need of ensuring semantic correctness, avoiding information loss, and guaranteeing an acceptable performance.

- Conceptually complex aggregations: The definition of summary tables in itself is a complex problem. For instance, generating aggregate data that allow rating workflow

resources has been subject of several investigations.

- Diversity and evolution management: Workflow models are continuously in evolution, and many products tend to add new features to the supported model. Although it is possible to focus the design on current models, even little changes to the workflow models might require a big warehouse redesign effort. In addition, the WDW should be able to import data from any WfMS.

Using workflow management systems improves the efficiency of business processes by automating the coordination of the data and resources used for the execution of differnt activities. Workflow management systems use a formal representation of the process logic, which is designed as a workflow model during the development phase of a workflow application. During the execution phase of the workflow application, the workflow engine derives workflow instances from the generic workflow model and notifies workflow participants about pending activities through their work lists.

7.2.2 Data Warehouse Workflow Management

Workflow data warehouse development is part of a bigger effort, purposing at developing a Business Process Intelligence (BPI) solution. The goal of BPI is to enable business and IT users to extract knowledge that is hidden in the WfMS logs and to be alerted of any critical situations or expected quality of service degradations. Another long-term goal is the ability of dynamically optimizing workflow definitions and executions. Details on the BPI effort are provided in [Casati 2001].

In the business process, events of interest are changes in the process and node execution states. Therefore, it is necessary to consider state changes as the facts of the data warehouse. WDW includes only facts about completed process instances, to simplify data archival and loading and to provide a simple framework in which data can be analyzed.

The structure and relationship definition of facts is complicated due to the variety of node types in most workflow models. For example, the HPPM process model includes a route and a work node to model routing decisions and service invocation, respectively. These nodes have many attributes that must be described in the WDW schema. For instance, a work node execution may be related to the invoked service or to the resource which executed the service, while a route node execution may be described by the set of arcs fired. Besides, different workflow models (or different versions of the same model) can have different types of nodes, as well as different attributes for a certain process. Therefore, we are faced with the problem of fact tables design so that all process and node facts can be considered, while easy maintenance and satisfactory performance are still enabled.

In this way, most reports and data aggregations can be computed on the basis of this table, which simplifies view definitions and avoids the need to join several tables for computing the results (see Figure 7.2 [Bonifati 2001]). Facts tables which are specific to a node type or to a process model may also be included in the data warehouse, thereby allowing the storage of attributes which are not included in the generic tables but can be required occasionally for specific reporting purposes.

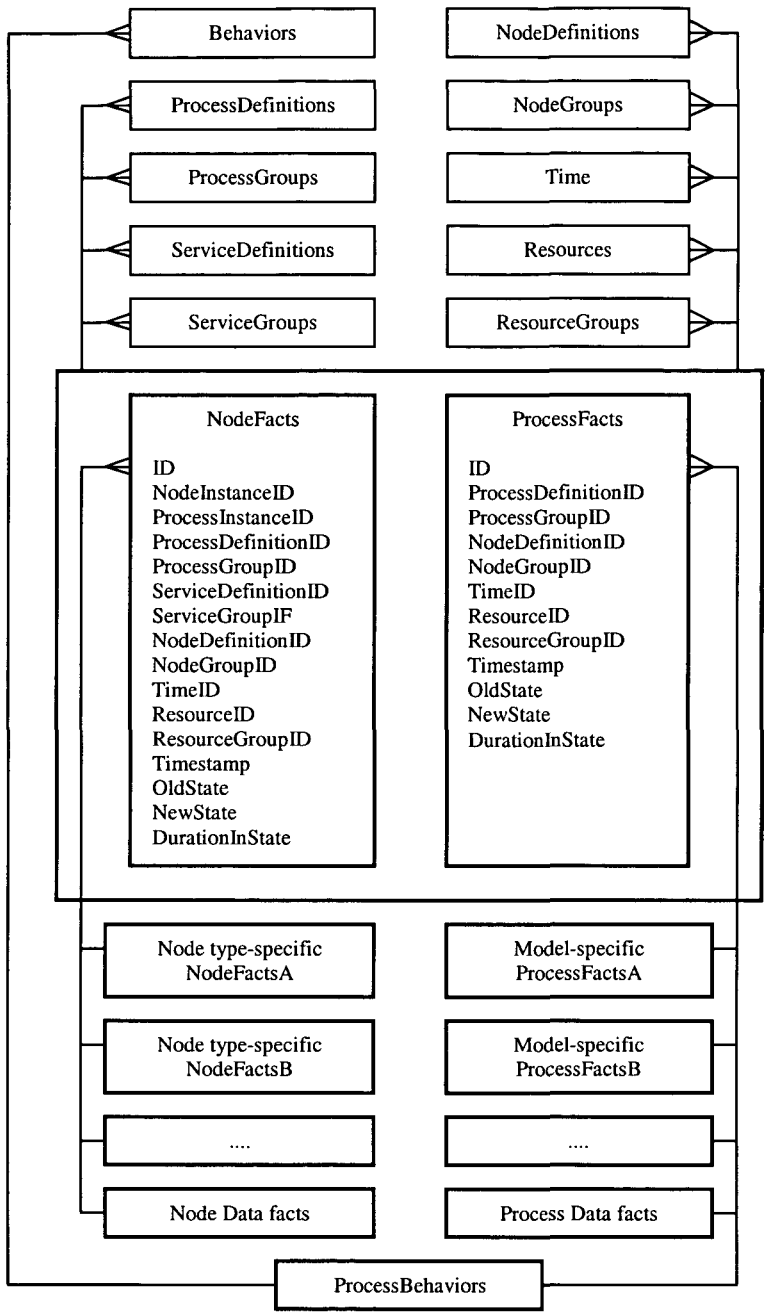


Figure 7.2: WDW schema. Facts are depicted with a thicker border, while dimensions have a thin border.

Another issue that must be handled is modeling data modifications. In fact, changes to process instance data may be of interest to the analyst. For example, analysts may want to view facts related to purchases of cars above $30,000. Data modifications are not part of the process and node fact tables. Actually, different processes and different nodes modify different data. Therefore, storing them horizontally in the relation is simply unfeasible. Instead, it is better to store data modifications in the ProcessDataFacts and NodeDataFacts relations, that include tuples describing the process or node state change in correspondence of which the data modification has occurred (e.g., a node completion), the name of the data item, and the new and old value.

Note that generally it is not necessary to log all data items. The analyst may be interested in only a fraction of them, and logging the complete data can be a very heavy burden for the warehouse. Therefore, WDW administrators can specify which data modifications are to be loaded, based on the data item name, and on the process and work node in the context of which the modifications take place.

The data extraction from workflow logs and the loading of the WDW can be performed by a set of ETL scripts provided with the WDW. It is assumed that log data are available as log files, extracted from WfMS logs (usually stored in relational databases). The first step is extracting data from the files and restoring the content in a relational format. After that, a sequence of cleansing operations can be performed. WDW provides a set of data checking and cleansing modules that process data without changing its structure. The advantage of this approach is that cleansing modules can be plugged in and out depending on the user's needs. Each cleansing module that is added causes delays in the load process, but guarantees data consistency, elimination of duplicates, and other advantages that are important for WDW integrity.

After that, data is inserted into WDW shadow database, i.e., a database with the same schema of WDW. Preparing the shadow tables instead of directly loading the warehouse has several motivations:

- Once the data is in the shadow tables, then the WDW can quickly be loaded using simple inserts or partition exchanges, reducing the WDW downtime.

- The schema of shadow tables is WfMS-independent, and thus they can be used to execute, WfMS-independent cleansing operations.

- Dimensions and relationships between facts and dimensions can be computed from the shadow tables. For instance, shadow tables can be used for collecting timestamps of facts, and load the time dimension table, to extract load statistics and to perform operations of higher-level, such as detecting behaviors.

Once cleansing and transformation operations have been completed, data are loaded into the WDW.

7.2.3 Process-driven Management Information Systems

Although workflow management systems can increase the process efficiency of an enterprise, they do not always lead to a more flexible organization. Due to the complexity and time-consuming nature of introducing and deploying a workflow-based information system architecture, it can be noticed, that once this architecture has been successfully deployed, many organizations resist the need to apply changes to the new system (an effect that can be observed at organizations introducing ERP packages as well).

However, the complexity of workflow projects is only one cause for this kind of attitude towards change management. An even more severe cause is the non-transparency of the relationships which describe the effects of changes to the workflow on the technical, organizational, or process level. This missing transparency can be related to the lack of an integrated infrastructure for gathering and presenting key performance indicators which

describe the behavior of a workflow-based organization and give advice on which parameters to change in order to increase the organizational efficiency.

During the execution phase of a workflow application, the workflow engine generates information about the different changes of state on the process and activity level that is loged in the audit trail, which is either file-based or a database structure. The audit trail is originally designed as a technical protocol for debugging purposes and provides information about the business processes execution at the operational level.

Theoretically, using audit trail information could enhance traditional enterprise controlling and management information systems as it presents key indicators about process performance, but also drill-down capabilities from single process instances to the business data related to these instances. In practice, the reporting components of many workflow management systems only offer limited evaluation functionality. For instance, the combination of workflow audit trail data with data warehousing technology is usually not supported by most workflow providers, but left to be extended by the end user.

7.2.3.1 Monitoring and Controlling in the Workflow Life Cycle

Companies need to quickly adapt to changing market conditions and customer needs. Having a good overview of ongoing and historical business processes makes companies flexible enough to adjust the treatment of individual cases, but also gives them the opportunity to make structural changes to business processes. Workflow management systems separate application logic from business process logic, and so they enable end users to modify the business processes on the basis of intelligence gathered from the use of a workflow application.

The workflow application life cycle can be described as a closed loop, starting with the definition of the business process that is to be implemented, followed by transforming the process model into a workflow model. Enacting the workflow model makes up the third phase of the workflow life-cycle, in which the analysis of executed workflows in the sense of process controlling generates information which is fed back into the process design phase. Other workflow cycles may contain different phases, but they all keep the development, deployment and analysis of a workflow application as a closed loop.

In practice, the assumption of closed loop does not reflect the development and deployment of workflow applications as it actually happens. The reason for this is that there is no isolated cause-effect relationship between the design and enactment of the workflow. Changes affecting the workflow performance can be applied not only to the workflow model itself, but also to the invoked application systems and to the organizational surroundings of the workflow application. If, for instance, the organizational signature power of certain workflow participants is increased, those participants could autonomously approve an increased number of cases, lowering the number of approvals needed by managers and this way reducing the workload of some resources. Although this measure has no direct effect on the workflow model or the workflow management system, it noticeably affects the process performance (e. g. lower average throughput time).

Introducing a workflow management system is a sequential process, which is similar to the development of a complex application system. The introduction of the system is followed by

cyclic workflow monitoring and controlling activities that run in parallel to system administration and maintenance tasks. The cyclic auditing of the workflow application is reflected by the organizational loop. Its feedback is applied in the technical loop by incrementally changing the workflow application. This way, it is possible to maintain a continuous improvement process, without facing the risks related to a redesign of existing processes and applications.

7.2.3.2 Process Monitoring

Process monitoring deals with the overview and analysis of process instances at run time. By the use of monitoring information, process managers and workflow administrators can adjust the behavior of existing workflow instances and react to problems that occur during process enactment. Moreover, process monitoring is used for improving the responsiveness of an organization to customer inquiries. When it is easy to determine the current state of a process instance, questions like "Who is handling the customer order number 57634?" can be answered more efficiently. For individual workflow participant, monitoring makes it possible to identify other employees who worked on a particular case before, in case of open issues which need to be resolved.

Process monitoring can also be used to predict staffing requirements. If the average processing time of activities allows a forecasting of open processes at a certain point in time, the number of active process instances and the current activities of these instances allow the short-term prediction of staff requirements. In combination with the ability of workflow management systems to prioritize work items according to the age of the case and the case attributes (e.g. the idle time of pending cases), process monitoring helps organizations to maintain a consistent level of cycle times even during seasons with higher workloads.

The importance of workload transparency can be demonstrated by an example of an insurance company. Due to a change of the tax legislature, life insurances had to undergo additional taxation, if the contract was signed on or after 1999. This led to a significant increase in life insurance applications by the end of 1999. The staff at the life insurance department worked hardly to handle the large amount of applications, disregarding all cases that were not new applications. This resulted in the structure and age of the remaining cases to be unknown, and customers complained about the very long time insurance took to contact them regarding their inquiries. This situation could have easily been avoided, if a workflow management system had kept track of all the cases and prioritized those older than a certain time.

Under certain circumstances, it is desirable not to expose the details of the process structure to the employees monitoring a certain workflow, for instance, the presentation of workflow data to workflow participants outside the company in which the workflow is executed, such as customers or suppliers. Figure 7.3 [Muehlen 2001] illustrates the abstraction level between the process state and a business state. The activities in section B and the activities in section C are combined into one status B' and C', respectively, while the activities A, D, and E have the same level of granularity in the business state model.

In this example, the business state model can contain only fewer or equal number of states than the process state model, as it is derived from the workflow states. If we take context data like the values of certain process relevant variables into account, the business state model states can be refined into sub-states.

In addition to the organizational process monitoring, workflow management systems usually offer technical monitoring, which deals with parameters like system load, response times, etc. Regarding technical monitoring workflow management systems typically are not significantly different from complex application systems managed through common commercial products. Other than the numbers of active users, activities, and processes, the system as well displays the number of pending activities and processes (e.g. the activities and processes which have been confirmed by a user but have not been processed completely).

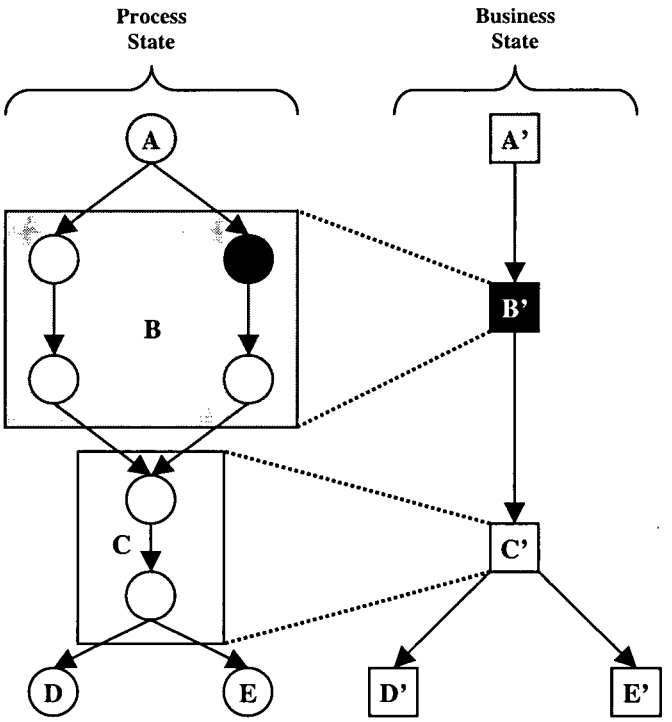


Figure 7.3: Process state and business state

7.2.3.3 Process Controlling

Process controlling mainly deals with the analysis of workflow audit trail data. The single instances in the workflow are aggregated according to different evaluation dimensions schemes. Workflow controlling can be used to detect long-term developments in workflow enactment and the review of existing workflow implementations. In order to identify abnormalities in the process execution, the audit trail data is usually compared to target data that is derived from corresponding business process models. The aim of workflow-based controlling is improving future process enactment, and therefore it has more long-lasting effects than the results of workflow monitoring. While the target group for workflow monitoring data is administrative personnel and workflow participants, workflow controlling data is used for organizational controlling purposes. Audit trail data when analyzed offers information related to temporal aspects of process execution, but also information about utilization of resources on the process and activity level. Still, information related to the business context of a certain process usually cannot be answered only by looking at audit trail data. This is because data that is processed in the applications invoked during workflow enactment is usually not stored by workflow management systems. The Workflow Management Coalition (WfMC) has published a definition of three classes of data associated with workflow systems [WfMC 1998]:

- "Application data is data beyond the control sphere of the workflow management system. It is managed and stored by the applications invoked during the enactment of workflows, e. g. a letter to a customer that is managed by the word processing system."

- "Workflow-relevant data is managed by applications and has an impact on the control flow of the current process. Typically this type of data is queried at decision nodes during the process, when the workflow management system has to decide which of several alternative paths to follow. Workflow-relevant data can also be used to increase the flexibility of staff assignment rules (e. g. "`IF claim.value()<50,000 THEN performer.role() = accountant ELSE performer.role() = manager`"). This type of data is read (but not updated) by the workflow management system, but only few systems store this information in their audit trail records."

- "Workflow-internal data is managed by the workflow management system itself and contains information about the current process instance, e. g. the ID of the process starter or the name of the performer of the last activity. This information is used to realize run-time specific semantics in the process flow, such as the assignment of an activity to the manager of the process starter. This is the kind of information found in most audit trail formats."

Many workflow management systems available today correspond to the WfMC separation of application data, workflow-relevant data and workflow-internal data. Also many systems allow the designer of the workflow to specify complex data structures and provide functionalities for specifying data flow and transformation. However, those systems usually do not store this data in the audit trail file.

From a controlling viewpoint, workflow audit trail data is another information source, just like log files from a transaction processing systems or financial statements from accounting systems. To make the audit trail data more valuable for business, it is necessary to combine it with application data. Data warehouses provide a repository for this kind of data, and there are many OLAP tools which support the controlling recipients during the evaluation of the information stored in a data warehouse. Hence, integrating the workflow audit trail data into the data warehouses provides a good opportunity to enhance controlling infrastructures with by analyzing the business process perspective of the enterprise.

For more details on data warehouse workflow management please refer to [Bonifati 2001] and [Muehlen 2001].

7.3 Example of Temporal Workflow Management

Temporal workflow management is necessary for time-driven processes. It deals to workflow management aspects related to the time dimension. Examples for that include the allocation of time to various workflow steps, the management of turnaround time of work, and the prioritization of tasks within the task queues of the workflow system.

[Zhao 1999] introduces a study of temporal workflow management in the context of claim handling that also discusses policy issues and business level concepts. Claim handling arises

in many organizations that provide services or sell products to their customers. It is an important business function since some products might fail and some customers might not be satisfied with the provided services. It is also significant as business context because it is somehow a complex process in which numerous types of services and products provided by the organization are found.

High quality claim handling is significant to keep customers satisfied and to maintain a solid market base. Moreover, claim handling has significant implications to engineering and manufacturing as it can resolve design and production problems in the organization. Despite the fact that claim handling has a special meaning in insurance companies (e.g. filing accident reports and authorizing compensation), [Zhao 1999] in their paper use this term to refer to the handling of customer complaints in any type of organization.

Current implementations in the management of claim handling are still very rudimentary. Claim handling processes are usually managed using first-come-first-serve (FCFS) queues. Some claims may be stamped as "Urgent" and thus they are given a higher priority. In some cases, the manager may push certain claims to get them processed faster. "Stamping" and "pushing" cases is so far the only way for the manager to control the claim handling workflows. Moreover, the manager has limited capability to predict and control the turnaround times. As level of workflow automation is increasing, it is possible to develop a solution for this problem.

7.3.1 Claim Handling System

Typically, a claim handling system is designed to provide a single entry point for customer complaints, product returns and repairs, and related business disputes. The claim handling process consists of many steps:

- Documentation: Customer complaints received through various means (e.g. emails, telephones, web pages, or walk-ins) are registered and documented.
- Classification and dispatching: Complaints are then classified according to the rules of the organization and dispatched to various groups and departments within the organization to resolve the problems.
- Diagnosis: Problems are then studied by technical experts to determine the reasons, solutions, responsibilities, and relevant business consequenses of the claim.
- Repair or replacement: Product problems must be fixed through repair or replacement. Certain repairs are done within the claim handling workshop, and other repairs are done either at the customers' sites for high value and heavy equipment or at the organization's premises.
- Testing and certification: Complex and mission-critical problems have to be tested after being resolved, and high cost problems have to be tested and certified by a special team.

- Settlement: The costs of repair and replacement must be debited to either the customer or the organization according to the organization's regulation and the guarantee terms.

- Dispute resolution: Very often, a dispute can arise as to who is at fault. Then, a higher level of organization authority or lawyers may need to get involved to handle the dispute and to reach an agreement.

- Monitoring and control: All processes have to be monitored and controlled by maintaining detailed records, alerting proper authorities if necessary, and following proper organization rules and approval channels.

Different claim cases take different routes through the workflow system and require different workflow steps. For example, some problems may require a return and refund, and the resulting workflow is fast and simple. Others may require a replacement. This is a little bit more complex than return and refund since some products may be out of stock and require a special production order; alternatively, they may be replaced with a different product. Problems requiring a repair imply more workflow steps. Simple repair work can be done at the claim handling department's workshop, but more extensive repairs may need to be sent back to the factory.

Different workflow steps result in all claims not having the same turnaround times. Claims necessitating an in-factory repair may need many days to be processed, which may cause customers to become impatient with the long time taken to complete the claim handling process. Moreover, customers are worried when the employees cannot predict reasonably precise how long the process might take. This is usually due to the traditional workflow processes usually being based on a FCFS policy because of a lack of temporal workflow functions in a manual workflow system. Workflow automation has brought up new opportunities for more complex time allocation and task prioritization policies.

7.3.2. Temporal Workflow Management Issues

[Zhao 1999] assume that "customer satisfaction depends on fast turnaround and thus concentrate on the reduction of turnaround time for claims requiring more complex processes". They also assume that "a workflow system is already in place so that many aspects of claim handling are done electronically":

- A modeling tool is used for specifying the workflow model which links all possible tasks in claim handling.

- Each worker has his work list, which contains a queue of tasks and their relevant documents.

- Workflow routing is usually done electronically based on the workflow model. That means, when a task is processed completely, the workflow system will automatically forward the information on subsequent tasks to the work lists following the current task.

- The workflow contains many decision nodes where decisions are made either by human agents or by software agents. This results in different paths for different

workflow cases.

- The workflow processes are automatically monitored and recorded, and the workflow system is capable to determine special cases and to alert the managers.

The turnaround times of claims for a defined set of system parameters (e.g. the number of workers, their role assignments, and the average work time for each task) can be managed as follows:

1. The expected turnaround time of a claim can be predicted using system parameters like the process model, the probabilities of branching at each decision node, the queue length at each node, and the mean work time for each task.

2. The expected process times of all tasks in the workflow can be allocated with the expected turnaround time. Since the different decision nodes in the process model may result in different paths, the number of tasks in each workflow case is unpredictable, and thus, techniques need to be used to determine times in the case of uncertainty.

3. Policies need to be developed for task prioritization for each work list. The prioritization can be done using some attributes of the tasks like the customer value factor, the expected completion time, the complexity of the workflow process, and the urgency value of the claim.

4. Because workflow systems are not always completely automated and usually workers can decide which task to be processed next, task prioritization cannot be mechanistically enforced as in machine scheduling. Hence, there should be a reward policy which encourages workers to follow the given priorities, but still allows them some autonomy.

Since claim process cases can take different paths with different results, the workflow system is unpredictable. As a result, it is necessary to estimate the expected turnaround time and allocate the total time to different tasks. These factors play a role in temporal workflow management:

1. Customers want to get fast results and want to be informed of the date of delivery right away,

2. Managers must balance the urgency and quality of processes, and

3. Employees need to know the expected deadlines to be able to prioritize their tasks.

In addition to time allocation and task prioritization, reducing turnaround times can also be achieved by adding new employees, reassigning roles to balance the workload, and employing better technology to reduce the process times of tasks. And as the system workload can be very high and the system can become overloaded, it is also important to consider system overloading problems.

7.3.3 Implementation of Temporal Workflow Management

"The temporal workflow management approach must be implemented through a workflow engine that understands the temporal policies and principles. This temporal workflow engine keeps track of the arrival of new claims and provides guidance to the workers in the workflow system. The workflow engine does so by computing various parameters and values needed by the time allocation and task prioritization policies". [Zhao 1999] describes a temporal workflow engine and the enactment procedure for temporal workflow management.

As previously mentioned, workflow systems are typically semi-automated, and many tasks in the system are performed either completely or partially by humans. Thus, the system cannot be scheduled as it is in a machine workshop where the subjects of discussion are numerically controlled machines. This means, the temporal workflow policies in a workflow system can only be provided as suggestions to the workers. Moreover, as the predicted turnaround times can never be exact and the time allocation schemes are also approximate, the may not always be accurate. As a consequence, it is preferred to give the workers the flexibility to deviate from the suggestions, and so providing system flexibility.

To offer support to workers on the temporal workflow policies, the workflow engine should be able to automate the computation process for the expected turnaround times, the time allocation principles, and the task prioritization policies. Figure 7.4 [Zhao 1999] shows the functions of the temporal workflow engine and the events and processes used to enact temporal workflow management.

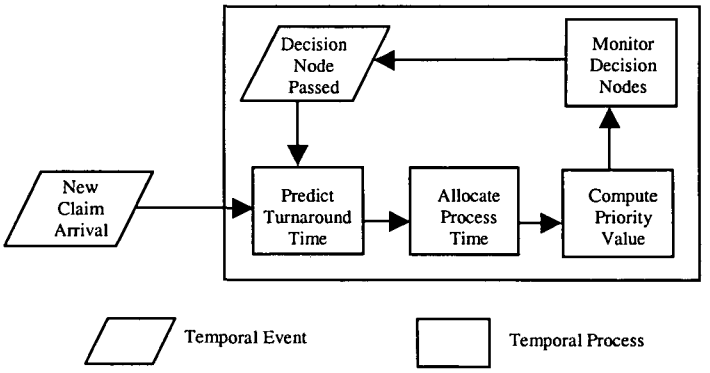


Figure 7.4: Temporal workflow engine functions

Claim handling is mainly a support function for the organization to provide after sales services, and its main goal is maintaining customer satisfaction. As a consequence, the quality of service is a very important part of the workflow process in claim handling. As previously mentioned, the use of a FCFS policy can cause long turnaround times for complex claims. Temporal workflow management provides a solution for this problem. Still, proper reward functions are needed to realize the business value of temporal workflow management. Examples of reward functions found in the industry are:

- Total business value delivered per unit time
- Number of tasks completed per unit time
- Number of claims completed per unit time.

Temporal workflow management enables a better control of the business processes, especially time-critical processes. With a systematic prioritization of workflow tasks, temporal workflow management can noticeably reduce the need for managers to "push" urgent claims personally. A better quality of service and an overall improvement in the business value of claim handling can be achieved by applying more sophisticated weight factors and reward functions.

For more details about temporal workflow management please refer to [Zhao 1999].

7.4 The Impact of Data Warehousing on Business Intelligence

The implementation a data warehouse infrastructure to support business intelligence (BI) bas become a big challenge in the last years with focus on the amount of failing data warehouse implementations – according to some reports approximately 85% of data warehouse implementations do not succeed.

There are several reasons for the failure of data warehouses in meeting the expectation of the enterprise, but one of the most significant reasons for failure is the lack of attention to the data. When data is considered it is usually only in terms of data integration (e.g. migrating data from different sources) rather than in terms of data quality. Moving data from its differnt sources into another repository is only one part of the challenge in delivering a data warehouse and BI. Without taking into account the accuracy, consistency, and timeliness of the data, business intelligence can lead to bad decision-making, higher cost, and lost opportunities.

According to industry analyst firm Gartner "more than 50% of business intelligence and customer relationship management deployments will suffer limited acceptance, if not outright failure, due to lack of attention to data quality issues. The impact of poor data quality is far reaching and its affects are both tangible and intangible. If data quality problems are allowed to persist, the executives grow to mistrust the information in the data warehouse and will be reluctant to use it for decision-making. Before long the data warehouse becomes another costly element that has failed to deliver benefit to the business".

But how can this problem be resolved, and who should be responsible within the organization? Data quality is a strategic issue which needs to be considered by the information consumers rather than the IT personnel. Finally, if a successful data warehouse implementation is desired, a high priority must be given to data quality, and data quality effort has to be recognized as an ongoing business issue in order to be successful.

Figure 7.5 shows how business-led data quality plays a significant role in supporting business intelligence. A big challenge for data architects is the increasing amount of operational data that has to be cleansed, integrated, and transformed. These tasks must deal with enterprise scale issues such as the variety of data, the increasing volume of data, and sometimes near real time data refresh levels.

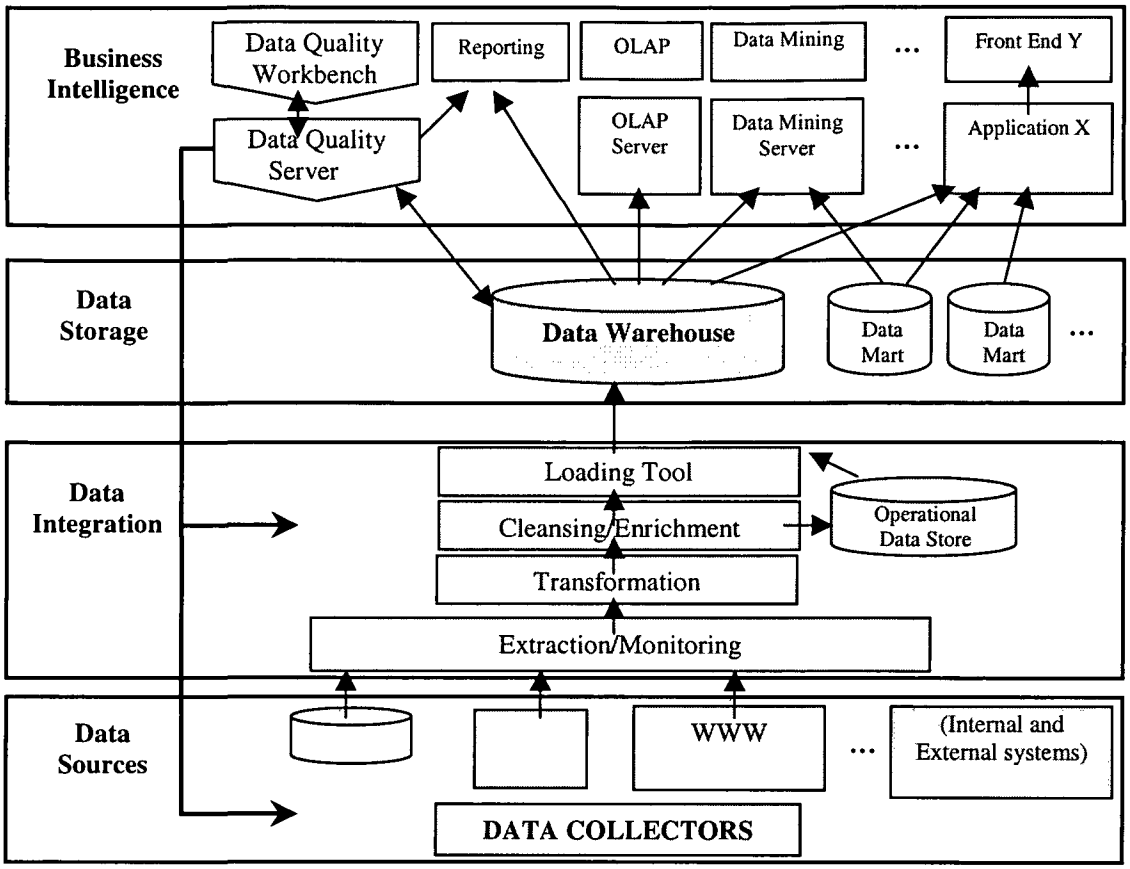


Figure 7.5: Business led data quality for data warehousing

The majority of data quality tools available today, no matter if implemented as a stand-alone solution or used to support ETL processing, are simply inadequate for enterprise-wide implementations. Only a few of them are able to scale to enterprise-level data volumes and refresh frequencies (e.g. from batch to continuous). For instance, let's look at the data flow that is dictated by the cleansing and transformation batch processing.

First, the data has to be extracted from its source and stored temporarily in tables or files to be processed. The data is then cleansed, transformed, or prepared otherwise according to pre-defined data quality rules. During this process, the data is moved in and out of temporary tables or files as the process may require. When the data meets the defined specifications, it is temporarily stored again. At the end, the data is loaded from the final temporary storage into the data warehouse tables or it is forwarded to the ETL tool to be processed. Considering all the batch movement of data which typical data quality software require, it becomes clear that the technology can easily be a process bottleneck with increasing data volumes or refresh rates.

Business intelligence provides a wide spectrum of analytical applications, from traditional data warehouse to "active" data warehousing to business activity monitoring. Between one application and the other, not only the source data may be different, but also the frequency of sourcing and the data detail and volume may be different.

Therefore, it is necessary to develop a formal data quality strategy and architecture, not just like the customary data architecture and technical architecture that exists in BI implementations. This architecture has to provide a simple integration of data quality metrics between different architectural points sharing the workload. This ensures scalability, regarding data volumes, frequency of data stream (continuous or batch), and complexity of processing.

For more details about data quality and business intelligence please refer to [English 1999].

CHAPTER 8

Conclusions

8.1 List of Conclusions

1. The most common issues related to representing time in the multidimensional data warehouse have been addressed and design techniques for implementing the time dimension have been introduced. Issues related to time dimension updates have also been covered with examples for structural and instance updates.

2. A data warehouse is a single, complete, and consistent store of data obtained from a variety of sources and made available to end users in a way they can understand and use in a business context. The goal of the data warehouse is to make accurate data, which is consistent across the enterprise, accessible to end-users in an efficient way, which is impossible when the data resides on an operational system.

3. Online Analytical Processing (OLAP) brings the ability for the user to think of data logically as multidimensional. It is the process of creating and managing multidimensional enterprise data for analysis and viewing by the user who seeks an understanding of what the data is really saying.

4. In the multidimensional model data is represented in logical dimensions in order to provide a consistent view of the data over time, with time being one of the major dimensions in every multidimensional data warehouse. The time dimension contains descriptive temporal information, and its attributes are used as the source of most of the temporal constraints in data warehouse queries. The design of the time dimension is not always straightforward as it strongly depends on the type of business and the requirements of the enterprise.

5. The time dimension can easily be built using a simple spreadsheet and can also be filled with a single SQL INSERT statement. However, problems will start to arise when the fact table requires granularity finer than a day, let it be an hour, a minute, or a second. For these cases the only way is to use SQL timestamps despite their limitations and to give up the ability to navigate through seasons and fiscal periods to the nearest second.

6. Holidays, seasons, and fiscal periods can be integrated into the time dimension in a way that effectively reduces query execution time and provides more functionality than using conventional RDBMS tables. This is done by adding new attributes and flags to the time dimension and using help tables to populate it with holidays, seasons, and fiscal periods for different countries or regions.

7. The granularity of the business facts can be increased either by increasing the granularity of the time dimension (e.g. from daily- to hourly-based), by adding timestamps to the fact table, or by using twin timestamps. The first timestamp would be an SQL TIMESTAMP and the second would be a day id, a foreign key connecting to a calendar day dimension.

This approach gives us the possibility to search for very precise time periods, but also navigate data, for instance, to see all transactions that occurred on a holiday.

8. The problem of observing daylight saving time (DST) has been solved by inserting 23 hour records into the time dimension on all days on which time is shifted to DST, and 25 hour records on days when time is set backwards. This approach can be implemented using universal time (UTC) or the local time of any time zone.

9. Handling different time zones can be enhanced by storing both: local and universal time. This can be done by using two dimensions for time: one for local and one for universal time. To add more flexibility and granularity the time-of-day can be separated from the day by using two dimensions: a date dimension and a time-of-day dimension for both local and universal time. This gives us altogether four dimensions for representing time.

10. Structural and instance updates to the time dimension must be handled differently than updates to other slowly changing dimensions. Common techniques for handling dimension updates have been introduced and applied with some modifications on the time dimension. Examples for structural and instance updates to the time dimension have been given, and an algorithm to perform them has been introduced supported by the SQL code for its implementation.

11. SAP Business Information Warehouse (BW) is a suitable and viable option for enterprise data warehousing. The information model of SAP BW has been examined, and previously introduced data warehouse concepts and techniques have been mapped to components of SAP BW. The representation of holidays, seasons, fiscal periods, DST, time zones in SAP BW is slightly different than most traditional data warehouse systems, but provides a lot of advantages to enhance business performance and support the global exchange of time-dependent business information and data archiving.

12. Data warehouse process and workflow management are being increasingly used by organizations to improve the efficiencies of their processes and reduce costs. The role of temporal aspects in data warehouse process and workflow management has been discussed and an example for the implementation of temporal workflow management for claim handling through a workflow engine has been presented. The impact of data warehousing on supporting business intelligence to meet the expectations of the enterprise has been addressed, and focus has been given to the lack of attention on data quality as one of the most significant reasons for the failure of data warehouse implementations.

8.2 Summary of Contributions

Here are the contributions of new knowledge this thesis has made. The thesis has:

1. Addressed the most common issues related to representing time in the multidimensional data warehouse and introduced simple and understandable design techniques using one or multiple time dimensions or time-stamps.

2. Demonstrated the representation of holidays, seasons and fiscal periods by extending the time dimension with new attributes and flags.

3. Introduced different approaches to increase the granularity of business facts up to the nearest second by using a combination of the time dimension and timestamps.

4. Addressed the issue of observing daylight saving time DST and how it affects the design of the time dimension, and provided an approach to handle this issue by introducing 23-hour and 25-hour days.

5. Modified this approach to handle different time zones by using multiple time dimensions and storing the universal time in addition to the user or customer's local time and separating the time-of-day in another dimension.

6. Presented examples of temporal queries in sales, financial, procurement, and healthcare applications, but also for different types of temporal methods, such as temporal projection, temporal slicing, and temporal join.

7. Addressed the most common issues related to handling time dimension updates by showing why the time dimension is different than other slowly changing dimensions, and introducing common techniques to handle dimension updates with respect to the nature of the time dimension.

8. Applied those techniques with some modifications on the time dimension to perform structural and instance updates.

9. Given examples of structural updates, like creating and deleting a hierarchy level and adding a new attribute or flag, and examples of instance updates, like setting an existing day to a holiday, changes to the fiscal periods, adding and deleting one or more years, and changing the DST switch days.

10. Presented an algorithm to perform these updates supported by the SQL code for its implementation, discussed general update issues and resolved them throughout these examples.

11. Addressed representing temporal information by using SAP BW as an enterprise data warehouse and investigated the information model of SAP BW with focus on the storage architectural layer and gave an overview of the time characteristics provided by SAP BW.

12. Introduced a mapping of temporal data warehouse concepts to SAP BW components like InfoCubes and master data tables, and showed how common business-related temporal issues, such as handling different time zones, representing holidays, fiscal periods and daylight saving time (DST) can be modeled using functions of SAP BW to improve the global exchange of time-dependent business information and data archiving.

13. Discussed the role of temporal aspects in data warehouse process and workflow management and the impact of data warehousing on supporting business intelligence to meet the expectations of the enterprise.

If this thesis has helped you to resolve even one significant issue or see things in a clearer way, then my efforts have been successful. Good luck!

8.3 Future Research

1. We showed how a data warehouse can use multiple timestamps and multiple time dimension keys. Sometimes it is necessary to even use multiple copies of the time dimension, for example to model time in different time zones without having to consider this in the application design. But sometimes a single time dimension can appear several times in the same fact table, for example if the fact table stores different dates, such as order date, packaging date, shipping date, etc. In this case we say that the time dimension has different "roles" in the warehouse. However, joining all these date keys to the same time dimension will not be straightforward as SQL will interpret them all as having to be the same date. Therefore, the fact table has to be joined with several copies of the time dimension, one copy for each date column in the fact table.

2. Some data warehouses also store data of mixed frequencies, such as data from a week and data from a month. Weekly and monthly data are incompatible and seem to cause many inconsistencies. Is there a systematic way of rolling-up weekly data to months? And do OLAP tools help to analyze cubes observed with different frequencies?

3. Dates and timestamps do not always represent just the occurrence of a single event or transaction. They can also represent the start and end of a time span that is of valuable meaning to the business, for example the start and end of a contract. In this case queries will always require comparing periods of time rather than simple timestamps. These periods are not always isolated, but might also overlap to make comparisons more complex [Kimball 2002] and [Snodgrass 2000].

4. To achieve interoperability between the different concepts of time a standardized semantic representation of the time issues described in this theis is necessary and desirable. As a consequence a representation of the time dimension in great detailedness in XML (e.g using XTM – the Topic Maps XML standard) is highly desirable and necessary and will build the continuation of the efforts described in this thesis. A DAML-OIL representation of those aspects is in the pipeline of our near future tasks. The precise semantic specification of business related time aspects (using XTM and DAML-OIL) is a goal of our work within the UN-WSIS (United Nations - World Summit on the Information Society) intentions towards interoperability and harmonization of IT-relevant global data. These efforts could be regarded as a principal continuation of the EDIFACT Time standardization initiative (e.g. the new XML representation of the DTM segment for "Time Zone Element").

References

[Allen 1983] Allen, J. F., 1983, *Maintaining Knowledge about Temporal Intervals*, Communications of the ACM, 26:832-843

[Bliujute 1998] Bliujute, R., Saltenis, S., Slivinskas, G., Jensen, C., 1998, *Systematic Change Management in Dimensional Data Warehousing*, Proceedings of the Third International Baltic Workshop on DB and IS, pp. 27—41

[Bonifati 2001] Bonifati, A., Casati, F., Dayal, U., Shan, M.C., 2001, *Warehousing Workflow Data: Challenges and Opportunities*, VLDB Journal, pp 649-652

[Bouzeghoub 1999] Bouzeghoub, M., Fabret, F., Matulovic, M., 1999, *Modeling Data Warehouse Refreshment Process as a Workflow Application*, Proceedings of International Workshop on Design and Management of Data Warehouses (DMDW'99), Heidelberg, Germany

[Bruckner 2001] Bruckner, R., Tjoa, A.M., 2001, *Managing Time Consistency for Active Data Warehouse Environments*, Proceedings of the Third International Conf on Data Warehousing and Knowledge Discovery (DaWaK 2001), LNCS 2114, pages 254--263, Munich, Germany, September 2001, Springer

[Casati 2001] Casati, F., Dayal, U., Grigori, D., Shan M.C., 2001, *Improving Business Process Quality through Exception Understanding, Prediction, and Prevention*, Proceedings of VLDB'01, Rome, Italy

[Codd 1993] Codd, E.F., 1993, *Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate*, E.F. Codd and Associates

[Demarest 1997] Demarest, M., 1997, *The politics of data warehousing*, http://www.hevanet.com/demarest/marc/dwpol.html

[Devlin 1997] Devlin, B., 1997, *The Data Warehouse from Architecture to Implementation*, Addison Wesley Longman, Inc.

[Eder 1987] Eder, J., Kappel, G., Tjoa, A.M., Wagner, R., 1987, *BIER - The Behavior Integrated Entity Relationship Approach*, in: S. Spaccapietra (ed.), Proceedings of the 5th Intenational Conference on Entity-Relationship Approach, North-Holland, Amsterdam

[Eder 2001] Eder, J., Koncilia, C., 2001, *Evolution of Dimension Data in Temporal Data Warehouses*, University of Klagenfurt

[Eder 2002a] Eder, J., Koncilia, C., 2002, *Representing Temporal Data in Non-Temporal OLAP Systems*, University of Klagenfurt

[Eder 2002b] Eder, J. Koncilia, C., 2002, *Incorporating ICD-9 and ICD-10 Data in a Warehouse*, University of Klagenfurt

[Egger 2004] Egger, N., 2004, *SAP BW Professional*, SAP Press

[English 1999] English, L.P., 1999, *Improving Data Warehouse and Business Information Quality: Methods for Reducing Costs and Increasing Profits*, John Wiley & Sons, Inc.

[Hezzah 2004a] Hezzah, A., Tjoa, A.M., 2004, *Design and Representation of the Time Dimension In Enterprise Data Warehouses - A Business Related Practical Approach*, Proceedings of International Conference on Enterprise Information Systems (ICEIS 2004), Porto, Portugal

[Hezzah 2004b] Hezzah, A., Tjoa, A.M., 2004, *Temporal Multidimensional Modeling with OLAP for Business Applications*, Proceedings of Business Information Systems (BIS 2004), Poznan, Poland

[Hezzah 2004c] Hezzah, A., 2004, *Modeling Temporal Characteristics with SAP Business Information Warehouse as an Enterprise Data Warehouse - A Business Performance Enhancing Practical Approach*, Proceedings of Conference on Information Science, Technology Management (CISTM 2004), Alexandria, Egypt

[Hezzah 2005] Hezzah, A., Tjoa, A.M., 2005, *Mapping Temporal Data Warehouse Concepts to SAP BW Components*, Submitted for Publishing at International Conference on Enterprise Information Systems (ICEIS 2005), Miami, USA

[Hurtado 1999] Hurtado, C., Mendelzon, A., Vaisman, A., 1999, *Maintaining Data Cubes under Dimension Updates*, Proceedings of the IEEE International Conference on Data Engineering

[Inmon 1996] Inmon, W., 1996, *Building The Data Warehouse*, John Wiley & Sons, Inc.

[Jarke 1995] Jarke, M., Gallersdörfer, R., Jeusfeld, M.A., Staudt, M., Eherer S., 1995, *ConceptBase - A Deductive Objectbase for Meta Data Management*, In Journal of Intelligent Information Systems, Special Issue on Advances in Deductive Object-oriented Databases, 4(2): 167-192

[Kimball 1996] Kimball, R., 1996, *The Data Warehouse Toolkit*, John Wiley & Sons, Inc.

[Kimball 1997] Kimball, R., 1997. *It's Time for Time*, DBMS Online

[Kimball 1998] Kimball, R., 1998, *The Data Warehouse Lifecycle Toolkit*, John Wiley & Sons, Inc.

[Kimball 1999] Kimball, R., 1999. *The Clickstream Data Mart in the Data Webhouse*, Intelligent Enterprise

[Kimball 2002] Kimball, R., 2002. *Tricky Time Spans*, Intelligent Enterprise

[Lang 1997] Lang, P., Obermair, W., Schrefl, M., 1997, *Modeling Business Rules with Situation/Activation Diagrams*, In: A. Gray, P. Larson (eds.): Proceedings of 13th International Conference on Data Engineering (ICDE '97), Birmingham, U.K., IEEE Computer Society Press

[McDonald 2002] McDonald, K., Wilmsmeier, A., Dixon, D. C., Inmon, W. H., 2002, *Mastering the SAP Business Information Warehouse*, John Wiley & Sons

[Mendelzon 1999] Mendelzon, A., Hurtado, C., Vaisman, A., 1999, *Updating OLAP Dimensions*, Proceedings of ACM-DOLAP'99

[Moody 2000] Moody, D., 2000, *From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design*, Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000)

[Muehlen 2001] Muehlen, M.Z., 2001, *Process-driven Management Information Systems - Combining Data Warehouses and Workflow Technology*, University of Muenster

[Nguyen 2000] Nguyen, T., Tjoa, A., Wagner, R., 2000, *An Object Oriented Multidimensional Data Model for OLAP*, Proceedings of 1st Int. Conf. on Web-Age Information Management (WAIM 2000)

[Pedersen 2001] Pedersen, P., 2001. *e-Decisions Transcript*, Norwegian School of Economics and Business Administration

[Prosser 2001] Prosser, A., Ossimitz, M. L., 2001, *Data Warehouse Management Using SAP BW*, UTB Stuttgart

[Ravat 2000] Ravat, F., Teste, O., 2000, *A Temporal Object-oriented Data Warehouse Model*, Proceedings of DEXA'00

[Shilakes 1998] Shilakes, C., Tylman, J., 1998, *Enterprise Information Portals*, Enterprise Software Team - available at http://www.sagemaker.com/company/downloads/eip/ indepth .pdf

[Snodgrass 2000] Snodgrass, R., 2000, *Developing Time-oriented Database Applications in SQL*, Morgan Kaufmann Publishers

[Thomsen, 1997] Thomsen, E., 1997, *OLAP Solutions*, John Wiley & Sons, Inc.

[Vaisman 2002] Vaisman, A., Mendelzon, A., Ruaro, W., Cymerman, S., 2002, *Supporting Dimension Updates in an OLAP Server*, Proceedings of CAiSE'02

[Vassiliadis 1998] Vassiliadis, P., 1998, *Modeling Multidimensional Databases, Cubes and Cube Operations*, Proceedings of 10th International Conference on Statistical and Scientific Database Management (SSDBM), Capri, Italy

[Vassilidis 2001] Vassiliadis, P., Quix, C., Vassiliou, Y., Jarke, M., 2001, *Data Warehouse Process Management*, Information Systems'01, 26, 8, pp 537-561

[WfMC 1998] Workflow Management Coalition, 1998, *Interface 1: Process Definition Interchange Process Model*, Document number WfMC TC-1016-P, Available at www.wfmc.org

[Wijsen 1999] Wijsen, J., Ng, R.T., 1999 *Temporal Dependencies Generalized for Spatial and Other Dimensions*, Proceedings of Spatio-Temporal Database Management

[Wijsen 2003] Wijsen, J., Bès A., 2003, *On Query Optimization in A Temporal SPC Algebra*, Data & Knowledge Engineering, Volume 44

[Yang 2000] Yang, J., Widom, J., 2000, *Temporal View Self-Maintenance in a Warehousing Environment*, Proceedings of EDBT'00

[Yu 1994] Yu, E., Mylopoulos, J., 1994, *Understanding 'Why' in Software Process Modelling, Analysis and Design*, Proceedings of 16th Inernational Conference on Software Engineering (ICSE), pp. 159-168, Sorrento, Italy

[Zhao 1999] Zhao, J.L., Stohr, E.A., 1999, *Temporal Workflow Management in a Claim Handling System*, Proceedings of International Joint Conference on Work Activities Coordination and Collaboration (WACC'99), ACM SIGSOFT Notes, Vol. 24, No. 2, pp. 187-195

Curriculum Vitae

Personal Details

Name: *Dipl.-Ing.* Ahmed Hezzah

Date of Birth: 23-11-1975, Bonn, Germany

Phone: +43- 51707- 46392 (Office)
+43- 676- 5888149 (Mobile)

E-Mail: ahmed.hezzah@siemens.com

Education

1994-1999 **VIENNA UNIVERSITY OF TECHNOLOGY** **Austria**
Master's degree in Computer Science with focus on "Information and Communication Systems". Master's thesis at SIEMENS AG Austria. Subject of the thesis "Porting C-Based Oracle-Applications from UNIX to Windows NT". Selected by the faculty as a student assistant for the lab "Database Systems"

1994 **GERMAN SCHOOL CAIRO** **Egypt**

Professional Experience

SIEMENS AG VIENNA **Austria**
Program and System Development

2002-2004 SAP Development and Consulting with focus on Customer Relationship Management CRM and Business Warehouse BW.
Project Management for test and acceptance. Operations conception and implementation of different Sales and Marketing scenarios in SAP R/3 and mySAP.com, maintenance of interface to Customer Master Data system, training support for end users and helpdesk employees, implementation of web reporting applications for BW.

1999-2002 Data Warehousing: System Design and Application Development.
Development of the application "Interchange Transaction Scheduler" used for contract management of energy-trading companies.
Bug fixes and enhancements for different data warehouse applications.
Customer support for many data warehouse projects worldwide.
Communication with Sun Microsystems in Germany.

1998 Master's thesis about "Porting C-Based Oracle-Applications from UNIX to Windows NT".

1996-1998 Total of 13 months of training at the department of Power System Control. Work on several projects like "SINAUT Spectrum Data Warehouse". Research and development of Oracle applications and user interface design for different applications.

	VIENNA UNIVERSITY OF TECHNOLOGY **Institute of Information Systems**	**Austria**
1997, 1998	Student assistant for the lab "Database Systems" for 2 continuous years.	

Project Experience

2004	**SIEMENS AG VIENNA** e.p@ss BW consulting and development. Enhancement and Redesign of the web reporting interface for Switzerland and development of parts of the BW data model and query definition.	**Austria**
2004	**SIEMENS AG MUNICH** Project management for acceptanece test of the mySAP CRM global corporate solution for sales and marketing "CONCORD". Definition of acceptance test concept and test plan. Creation of test scenarios and test cases. Organiziation of the acceptance test performance and generation of acceptance reports.	**Germany**
2002-2003	**INFINEON TECHNOLOGIES AG MUNICH** mySAP CRM development and consulting with focus on Opportunity and Activity Management, Internet Sales, Marketing, Middleware, Mobile Sales and Service. Operation conception and implemention in SAP R/3 and mySAP.com. Maintenance of interface to Customer Master Data system. Training support for end users and helpdesk employees.	**Germany**
2003	**SIEMENS AG MUNICH** Concept for a mySAP CRM worldwide solution for sales and marketing. Definition of the CRM system configuration with backbone Spiridon and ep@ss.	**Germany**
2002	**SIEMENS AG VIENNA** Concept for a mySAP CRM solution for Sales, Marketing, Business Partners, Products, Organizational Structure, User Interface, Customizing) for selected fields of SIEMENS AG Austria.	**Austria**
1999-2002	**EnBW STUTTGART** **Germany** **EZH ROTTERDAM** **Netherlands** **BEWAG BERLIN** **Germany** **TEAS ANKARA** **Turkey** **EDISON MILAN** **Italy** **NWS STUTTGART** **Germany** System design and application development for the SINAUT Spectrum Data Warehouse using different development environments, such as FORTÈ/TOOL, C, C++, PL/SQL. Development of the application "Interchange Transaction Scheduler" used for contract management of energy-trading companies. Bug fixes and enhancements for different data warehouse applications. Customer support for many other data warehouse projects worldwide.	

Technical Know-How

Operating Systems:	Windows, Unix (Linux, Sun Solaris), pSOS+
Programming Languages:	ABAP, C, C++, TOOL, Java, JavaScript, SQL, HTML, XML, WML, Turbo Pascal, Prolog, Sun XDR, Modula-2
Database Systems:	ORACLE, SQL Server, Sybase, Informix, MySQL
Applications & Technologies:	Forté, Visual C++, Delphi, Database Deign, Web Development, Object-oriented Programming, UML
SAP:	R/3, CRM, BW, PLM

Courses and Conferences

	ANCIENT LIBRARY OF ALEXANDRIA	**Egypt**
Jul 2004	Conference on Information Science, Technology Management	
	POZNAN UNIVERSITY OF ECONOMICS	**Poland**
Apr 2004	International Conference on Business Information Systems	
	UNIVERSIDADE PORTUCALENSE PORTO	**Portugal**
Apr 2004	International Conference on Enterprise Information Systems	
	SAP AG VIENNA	**Austria**
Jan 2002	mySAP.com CRM Fundamentals	
Feb 2002	mySAP.com CRM Mobile Sales and Service	
	SUN MICROSYSTEMS SAN FRANCISCO	**USA**
May 1999	Forté Application Development	
May 1999	Forté Object Oriented Analysis and Design	
Jun 2000	JavaOne Conference	
Apr 2001	Forté Advanced TOOL Client Programming	
Apr 2001	Forté Performance and Patterns	
	ORACLE GmBH VIENNA	**Austria**
Jun 1999	Professional Introduction to Oracle SQL	
Oct 2001	PL/SQL und Database Programming	
	SIEMENS AG VIENNA	**Austria**
Mar 1999	Data Warehouse Basis	
Jul 1999	Data Warehouse Administration	
Oct 1999	Clear Case Basis	
Mar 2000	Spectrum Basis	
Mar 1999	SEM-System Development Method	
Nov 1999	SEM-Software Intensive Inspection	

Research

Since 2000 **VIENNA UNIVERSITY OF TECHNOLOGY** **Austria**
Institute of Software Technology and Interactive Systems
PhD in Computer Science about "Modeling Time in Temporal Multidimensional Data Warehouses" with focus on "Implementing an Optimized Time Dimension for Data Warehouses". Expected end in October 2004.

Member of the "Institute for Systems and Technologies of Information, Control and Communication INSTICC" and the "Austrian Computer Society OCG".

Publications

Hezzah, A., Tjoa, A. M., "Design and Representation of the Time Dimension In Enterprise Data Warehouses - A Business Related Practical Approach", In Proc. of ICEIS'04, 2004

Hezzah, A., Tjoa, A. M., "Temporal Multidimensional Modeling with OLAP for Business Applications", In Proc. of BIS'04, 2004

Hezzah, A., "Modeling Temporal Characteristics with SAP Business Warehouse as an Enterprise Data Warehouse", In Proc. of CISTM'04, 2004

Hezzah, A., Tjoa, A. M., "Mapping Temporal Data warehouse Concepts to SAP BW Components", Submitted at ICEIS'05, 2005

Technical Points of Interest

Data Warehousing, Business Intelligence, Temporal and Multidimensional Databases, Customer Relationship Management CRM, Object-oriented Analysis & Design, Web-Development, E-Business, E-Commerce, Application Development for Mobile Devices.

Languages

Native Language: Arabic

Foreign Languages German (fluently)
English (fluently)
Italian (good)
Spanish (basic knowledge)

Vienna, 01.09.2004