

Master's Thesis:  
Application  
of Fuzzy Classification  
to a Data Warehouse  
in E-Health

Professor: Prof. Dr. Andreas Meier  
Supervisor: Christian Mezger  
Information Systems Research Group  
Department of Informatics  
University of Fribourg, Switzerland  
<http://diuf.unifr.ch/is>

Student: Michael Kaufmann  
Route de la Pisciculture 19  
1700 Fribourg  
[michael.kaufmann@unifr.ch](mailto:michael.kaufmann@unifr.ch)  
<http://homeweb2.unifr.ch/kaufmanm/pub/>

April 21, 2006

## Abstract

The ability to analyze large amounts of data for the extraction of valuable information presents a competitive advantage for any organization. The technologies of data warehousing, OLAP, and data classification support that ability. The data warehouse is a central data pool which integrates heterogeneous data sources and provides strategic information for analysis and decision support. OnLine Analytical Processing (OLAP) presents an approach to data analysis where data is consolidated and aggregated with respect to multiple dimensions of interest. The idea is to consolidate large amounts of data by summarizing and aggregating data elements for every cell of a data cube. Classification of data elements reduces an arbitrarily high number of data elements into an arbitrarily small set of classes, which highly reduces the granularity of data. In OLAP, classification is used for the consolidation of dimensional attributes. Fuzzy classification is achieved by modeling data element classes with fuzzy sets. Thus, data elements are assigned to one or many classes to a certain degree.

This thesis researches the application of fuzzy classification to OLAP data analysis. Specifically, the approach is the fuzzification of dimension classification in OLAP cubes. A framework has been developed, which allows the description of data cubes with fuzzy dimension hierarchies. This framework has been implemented in a prototype computer program capable of calculating and manipulating OLAP cubes with fuzzy dimension classification. This prototype allows the definition of fuzzy contexts on dimensional attributes, as well as the definition of OLAP cubes on base relations. The levels of consolidation for every dimension can be drilled down, rolled up, sliced and diced. Data consolidation can be achieved by crisp or fuzzy classification.

**Keywords:** *Data warehouse, fuzzy classification, e-Health, OLAP, multidimensional data analysis, fuzzy data consolidation, fuzzy contexts, fuzzy data aggregation.*

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Problem Statement . . . . .	1
1.2	Thesis Objective . . . . .	2
1.3	Approach . . . . .	2
<b>I</b>	<b>Technological Background</b>	<b>4</b>
<b>2</b>	<b>Data Warehouse Systems</b>	<b>5</b>
2.1	Definition of the Data Warehouse . . . . .	5
2.1.1	Inmon's Definition . . . . .	5
2.1.2	Lehner's Definition . . . . .	6
2.2	Data Warehouse System Components . . . . .	6
2.3	OLAP . . . . .	8
2.3.1	Definition of OLAP . . . . .	8
2.3.2	Multidimensional Data Analysis . . . . .	9
2.3.3	OLAP Operators . . . . .	11
2.3.4	OLAP Implementation . . . . .	11
2.4	Data Warehousing Technology . . . . .	12
2.4.1	SQL Extensions . . . . .	12
2.4.2	MDX . . . . .	13
2.4.3	CWM . . . . .	13
2.4.4	JOLAP . . . . .	14
<b>3</b>	<b>E-Health and Healthcare Information Systems</b>	<b>15</b>
3.1	Definition of E-Health . . . . .	15
3.2	Fields of Application for E-Health . . . . .	15
3.3	Data Warehousing and E-Health: The Clinical Data Warehouse . . . . .	17
3.3.1	Case Study: A Clinical Data Warehouse Implementation . . . . .	18
3.4	Ethical Issues . . . . .	18
3.5	Legal Issues . . . . .	18
<b>II</b>	<b>Theoretical Background</b>	<b>20</b>
<b>4</b>	<b>Epistemology of Information Systems</b>	<b>21</b>
4.1	Knowledge . . . . .	21

---

4.1.1	Propositional Calculus . . . . .	22
4.1.2	Predicate Calculus . . . . .	22
4.2	Information . . . . .	24
4.3	Data . . . . .	24
4.3.1	Relational Data Model . . . . .	25
4.3.2	Multidimensional Data . . . . .	26
<b>5</b>	<b>Fuzzy Set Theory and its Applications</b>	<b>29</b>
5.1	Fuzzy Sets . . . . .	29
5.2	Operators on Fuzzy Sets . . . . .	30
5.2.1	Algebraic Fuzzy Set Operators . . . . .	30
5.2.2	A Compensating Fuzzy Set Connector . . . . .	31
5.3	Fuzzy Logic . . . . .	33
5.3.1	Linguistic Variables . . . . .	33
5.3.2	Fuzzy Propositions and Approximate Reasoning . . . . .	34
5.4	Possible Applications of Fuzzy Sets to a Data Warehouse . . . . .	35
5.4.1	Fuzzy Multidimensional Data Bases . . . . .	35
5.4.2	Fuzzy Data Aggregation . . . . .	35
<b>6</b>	<b>Fuzzy Context Classification</b>	<b>37</b>
6.1	Fuzzy Classification . . . . .	37
6.2	Mathematical Concepts used for Context Classification . . . . .	38
6.3	Context Classification . . . . .	38
6.4	Fuzzy Context Classification . . . . .	40
<b>III</b>	<b>Fuzzy Classification in Online Analytical Processing</b>	<b>44</b>
<b>7</b>	<b>A Framework for Fuzzy Data Consolidation</b>	<b>45</b>
7.1	Multidimensional Information . . . . .	46
7.2	Fuzzy Categorization . . . . .	48
7.3	Fuzzy Multidimensional Tuple Classification . . . . .	50
7.3.1	Tuple Classes of Basic Granularity . . . . .	50
7.3.2	Categorized Tuple Classes . . . . .	51
7.3.3	Fuzzy-Categorized Tuple Classes . . . . .	52
7.4	Fuzzy Data Aggregation . . . . .	53
7.5	Fuzzy OLAP Cubes . . . . .	55
7.5.1	OLAP Cubes with Crisp Dimension Categorization . . . . .	56
7.5.2	OLAP Cubes with Fuzzy Dimension Categorization . . . . .	57
7.5.3	Summarizability . . . . .	60
7.5.4	Operators on Fuzzy OLAP Cubes . . . . .	61
<b>8</b>	<b>A Prototype of an OLAP-Tool Supporting Fuzzy Consolidation</b>	<b>62</b>
8.1	Implementation . . . . .	62
8.1.1	Functionality Specification . . . . .	62
8.1.2	System Architecture . . . . .	63
8.1.3	Metadata . . . . .	63
8.1.4	Implementation of Fuzzy Consolidation . . . . .	65
8.2	User Manual . . . . .	67

---

8.2.1	System Installation . . . . .	67
8.2.2	Data loading . . . . .	68
8.2.3	Overview . . . . .	68
8.2.4	Data Administration . . . . .	69
8.2.5	Data Navigation . . . . .	72
8.2.6	Data Input and Output . . . . .	76
8.3	Evaluation and Outlook . . . . .	77
<b>9</b>	<b>Conclusion</b>	<b>80</b>
9.1	Summary of Results . . . . .	80
9.2	Personal Evaluation . . . . .	81
9.3	Outlook . . . . .	81
	<b>References</b>	<b>83</b>

# List of Figures

2.1	Components of a Data Warehouse System (Lehner, 2003)	7
2.2	<i>Multidimensional Information on Patients</i>	10
2.3	A Consolidation Path for Attribute <code>TimeOfVisit</code>	10
2.4	<i>The Star Scheme: Fact Tables and Dimension Tables</i>	12
3.1	<i>Role of the electronic health card in integrated healthcare</i>	16
4.1	<i>Semantics of the Predicate Calculus</i>	23
5.1	<i>Minimum Intersection</i>	30
5.2	<i>Maximum Union</i>	31
5.3	<i>Algebraic Product Intersection</i>	32
5.4	<i>Algebraic Sum Union</i>	32
5.5	<i>Gamma Operator: <math>z = \Gamma(x, y), \gamma = 0.5</math></i>	33
5.6	<i>Membership Functions for Terms of a Linguistic Variable</i>	34
6.1	<i>Example Contexts Defined on the Domain of the Attributes of Relation 'patient' (Table 4.1)</i>	39
6.2	<i><math>f\_level(bmi)</math>: A Fuzzy Context on Attribute 'bmi'</i>	42
6.3	<i><math>f\_health(bmi)</math>: A second Fuzzy Context on Attribute 'bmi'</i>	42
6.4	<i><math>f\_level(cstrel)</math>: A Fuzzy Context on Attribute 'cstrel'</i>	43
6.5	<i><math>f\_level(d\_bp)</math>: A Fuzzy Context on Attribute 'd_bp'</i>	43
7.1	<i>A framework for fuzzy data consolidation</i>	45
7.2	<i>A Point Lattice on <math>\mathbb{R}^2</math></i>	47
7.3	<i>Multidimensional Information</i>	48
7.4	<i>Categorization of a Domain</i>	48
7.5	<i>Classification of Tuples based on Attribute Values</i>	51
7.6	<i>Classification of Tuples based on Attribute Categories</i>	52
8.1	<i>FC-OLAP: Functionality Overview</i>	63
8.2	<i>FC-OLAP System Architecture</i>	64
8.3	<i>Metadata Sturcture Diagram</i>	65
8.4	<i>FC-OLAP User Interface Look and Feel</i>	68
8.5	<i>The Admin Menu</i>	69
8.6	<i>Establishing a Database Connection</i>	69
8.7	<i>Adding A Relation</i>	70

---

8.8	<i>Defining a Cube</i>	70
8.9	<i>Defining a Context</i>	71
8.10	<i>Defining an Interval for a Context Class</i>	71
8.11	<i>Defining a Fuzzy Context Class</i>	72
8.12	<i>Defining a Fuzzy Context Class2</i>	72
8.13	<i>The Data Menu</i>	73
8.14	<i>Creating a Cube View</i>	73
8.15	<i>Resulting Cube View</i>	74
8.16	<i>Slicing a Cube's Dimension</i>	75
8.17	<i>Cube Resulting From a Slice Operation</i>	75
8.18	<i>Dicing a Cube</i>	76
8.19	<i>Cube Resulting From a Dice Operation</i>	77
8.20	<i>Changing a Cube Dimension's Consolidation Level</i>	77
8.21	<i>Cube Resulting From a Drill Operation</i>	78
8.22	<i>The File Menu</i>	78

# List of Tables

- 4.1 *Relation 'patient', used as example throughout the paper . . . . .* 26
- 7.1 *Example categorizations of the attributes of relation 'patient' . . . . .* 49
- 7.2 *Example cube on relation 'patient' with crisp dimension categorization . . . . .* 57
- 7.3 *Example cube on relation 'patient' with fuzzy dimension categorization . . . . .* 59

# Chapter 1

## Introduction

Accurate information about an organization's state is necessary in order to make strategic decisions. An increasing number of heterogeneous information systems and an expanding data volume makes retrieving meaningful information more difficult. The data warehouse concept (Inmon, 1996) is an information systems approach for decision support which focuses on the integration of heterogeneous data sources in order to facilitate the derivation of decision support information. (Codd, 1993) proposes a similar solution, which is named Online Analytical Processing (OLAP). "Business enterprises prosper or fail according to the sophistication and speed of their information systems, and their ability to analyze and synthesize information using those systems." The OLAP concept suggests multidimensional analysis on integrated and consolidated data in order to accomplish that objective.

### 1.1 Problem Statement

Healthcare organizations use information systems as well as commercial companies. In the same way, they struggle with a growing data volume and a difficulty in making use of the stored data. However, decision support information for healthcare is usually less centered on increase of financial benefit. What is more important is the reduction of cost and the improvement of quality (Schilp & Gilbreath, 2000). E-Health is a relatively new term in healthcare information systems. It is often used to describe internet-based healthcare applications (Warda & Noelle, 2002), or generally the use of information technology for the improvement of healthcare quality (Jähn & Nagel, 2003). A data warehouse in an e-Health environment can enable healthcare organizations to make information accessible and analyzable in order to provide a basis for their decision support.

The main question of this thesis is "how can fuzzy classification be applied to a data warehouse, and how can such an application be helpful in order to improve decision support information for healthcare?" If the data warehouse is a system together with processes and applications, the question is where fuzzy classification can be applied *within that system*.

This thesis examines the applications of fuzzy classification in OLAP data analysis in a data warehouse. A way of retrieving meaningful information from large data pools is summarization by information clouding, which is accomplished by fuzzy classification of data. (Shenoi, 1995) uses classification for information clouding in database security. In data analysis, classification can be used to reduce the information volume by increasing its density. Thus, information clouding can hide unwanted details to facilitate decision support by summarizing and aggregating data in an information system.

In OLAP, data are consolidated along multiple data consolidation paths called *dimensions*. Data

consolidation paths can be expressed by context classification hierarchies on dimensional attributes. In crisp classification, information clouds are sets of data elements. In fuzzy classification, these clouds are described by fuzzy sets. Using fuzzy classification, different levels of dimension classification form a *fuzzy data consolidation path*.

## 1.2 Thesis Objective

The main thesis objective is to show a possible application of fuzzy classification to OLAP by designing a framework for fuzzy data consolidation, and by implementing a prototype based on the proposed framework. As stated in (Meier *et al.* , 2003), we assume that using linguistic variables, an analysis tool can provide a more natural classification of data in a data warehouse. Thus, the objective is (1) to provide a framework for the use of linguistic variables and fuzzy classification in OLAP cubes, and (2) to demonstrate the framework's feasibility by the implementation of a prototype OLAP application supporting fuzzy data consolidation.

The framework proposes five levels of data consolidation, as illustrated in Figure 7.1. On the first level, there is the basic data stored in a base relation. On the second level, the dimensional attributes of that relation are categorized using fuzzy contexts. On the third level, tuples of the base relation are classified using the dimension categories as classification features. On the fourth level, the tuple classes are aggregated using standard aggregation functions. On the top level, these aggregated data element classes are presented in a fuzzy OLAP cube, which provides a multidimensional access to all aggregated classes together with the possibility of manipulating the cube content.

Accordingly, the thesis objective is the description and implementation of every component of the framework. Thus, the thesis objectives are the *design of a framework* and a corresponding *software implementation* for each of the following points:

1. Multidimensional information: The data domain in which the analysis takes place.
2. Fuzzy categorization: Assigning dimension values to categories based on fuzzy contexts.
3. Fuzzy classification: Assigning tuples to classes based on the dimension value categories of a tuple. In an OLAP cube, there is one class for every combination of dimension values.
4. Fuzzy aggregation: Summarizing attribute values of data elements in a fuzzy class using an aggregation function.
5. Fuzzy OLAP cubes: Layout, navigation and manipulation of those aggregated classes.

## 1.3 Approach

Our approach proposes the application of fuzzy classification to data consolidation in multidimensional data analysis. Specifically, dimension value classification is fuzzified. In fact, dimension categories can be expressed by a classification of the dimension's domain through fuzzy contexts.

The approach will fulfill the thesis objectives by providing a theoretical framework and a practical implementation of each of the five points outlined by Figure 7.1. The general framework for fuzzy data consolidation will be presented in Chapter 7, whereas the implementation of a prototype is described in Chapter 8

1. Multidimensional information: In Section 7.1, the concept of information space is defined by the Cartesian product of orthogonal attribute domains.

2. Fuzzy categorization: In Section 7.2, fuzzy dimension categorization is described by fuzzy partitions of the dimension's domain. A fuzzy context class will model a dimension value category.
3. Fuzzy classification: In Section 7.3, fuzzy classification is described as the process of assigning data elements to fuzzy sets by using fuzzy dimension value categories as classification features.
4. Fuzzy aggregation: In Section 7.4, the standard relational aggregation functions are fuzzified in order to apply them to fuzzy sets of tuples.
5. Fuzzy OLAP cubes: In Section 7.5, fuzzy OLAP-cubes are described as a collection of data cells. There is one data cell for each combination of dimension values. Every data cell corresponds to a class of tuples, where the dimension values are the classification features. A fact in a data cell of a cube presents an aggregation of the measure attribute for each tuple in the corresponding class. Furthermore, a set of basic operators for the manipulation of consolidation paths is proposed.

The thesis is structured in three main parts. In the first part, the technological background is presented. A section on data warehouses discusses two definitions of the term and gives an overview of system components, processes and information structures described by (Lehner, 2003). A second section on e-Health contains some background, examples and issues about the relatively new topic, and discusses an existing implementation of a clinical data warehouse. The second part summarizes the theoretical state of the art of data models, fuzzy sets and their applications, and fuzzy context classification. In the third part, the main thesis approach is presented. A framework for multidimensional fuzzy classification for OLAP-cubes is formally developed. In a last chapter, the feasibility of the application of fuzzy classification to OLAP-cubes is shown by an implementation of a prototype OLAP application.

## **Part I**

# **Technological Background**

## Chapter 2

# Data Warehouse Systems

In this chapter, the concept of data warehouse systems is elaborated. The origins as well as current developments in the field are presented. In the first section, definitions of the data warehousing concept of (Inmon, 1996) and (Lehner, 2003) are compared. In the second section, the components of a data warehouse system, its applications and processes, as described by (Lehner, 2003), are summarized. Third, an overview of OLAP and multidimensional analysis is given. The last section mentions four recent technological developments in the field.

### 2.1 Definition of the Data Warehouse

What is a data warehouse? Bill Inmon is cited very often and seems to be the father of the term. In fact, Inmon's definition goes back to the first edition of his book "Building The Data Warehouse" from 1993. Probably, the article before the noun is deliberately used in the specific form, in order to emphasize the fact that there is supposed to be only one data warehouse in an organization.

Wolfgang Lehner, a researcher in data warehousing, has recently published a profound and comprehensive book on data warehouse systems in German (Lehner, 2003). He references Bill Inmon, but his book also contains a more elaborate definition of data warehouse systems, which will be presented in the section on data warehouse system components.

#### 2.1.1 Inmon's Definition

According to Bill Inmon (Inmon, 1996), a data warehouse is a "subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decision support process." Inmon describes the components of his definition in the following way:

- *Subject-oriented:* Operational databases are organized around applications of the company. Contrarily, the data warehouse is oriented towards the major subject areas of the corporation that have been defined in the data model.
- *Integrated:* In a large corporation, data is usually stored in many different legacy databases. Often, there is no consistency in encoding and data types. The data warehouse contains data that has been cleansed of all inconsistencies stemming from heterogeneous operational applications.
- *Nonvolatile:* In operational databases, records are regularly updated and manipulated. In contrast, data that has been loaded into the data warehouse can only be accessed for reading.

- *Time-variant:* The content of the data warehouse grows over time, where on regular basis a snapshot of current data is entered into the data pool. The key structure of the data warehouse always contains time.
- *Collection of data in support of managements decision support process:* Being a decision support system, the data warehouse is about going from data to information. It's about providing the management with valuable information upon which critical decisions can be based. The data warehouse integrates the corporation's data in order to facilitate the creation of valuable decision support information.

According to Inmon, the most important design issue for the data warehouse is granularity, the level of detail in the data. The more data is summarized, the coarser is the granularity. Through granularity, the volume of data is traded off against the possible level of detail of a query.

### 2.1.2 Lehner's Definition

Based on Inmon's definition, Lehner has introduced an extended version of that definition in (Lehner, 2003). He defines a *data warehouse system* as a collection of system components and data bases which satisfy the following conditions:

- *Analysis-oriented data organization:* Data organization is logically and physically directed toward analysis. This organization is preferably multidimensional, where quantifiable facts are distinguished from descriptive dimensions.
- *Integration of data from heterogeneous source systems:* Data is extracted from different sources, cleansed, and collected in an integrated data basis.
- *No updates by users:* Users of a data warehouse system can access this system for reading only. The data base is periodically filled with new data from external source systems by an administrator.
- *Optional historization of data:* Data that has been written into a data warehouse system once is not to be overwritten, but extended. Thus, a historization of states of the source systems emerges over time. Lehner writes that this historization, especially one with explicit temporal model support, is often not implemented due to limited memory space.

This definition differs from Inmon's definition in two main points. First, for Lehner, the temporal component is optional, whereas for Inmon, the collection of snapshots of the source systems, the historization, is necessary. Second, Lehner assumes the multidimensional organization of data in the data warehouse. In Inmon's sense however, data in the data warehouse is not necessarily multidimensional. In order to provide data for OLAP, Inmon proposes specialized application databases called data marts, which are specialized multidimensional databases for data analysis.

Lehner as well as Inmon stress the aspect of integration as the most important purpose of a data warehouse. Additionally, Lehner writes that a data warehouse would not even be necessary if the company-wide data model was implemented correctly on interoperable databases, because there is no need for integration if there is no inconsistency or heterogeneity in data storage systems.

## 2.2 Data Warehouse System Components

Lehner describes the systems and processes that belong to a data warehouse. A data warehouse system consists of up to six components. Data in a data warehouse system is processed by each of these

components. Figure 2.1 illustrates this data processing, where the arrows represent the flow of data elements between the different system components.

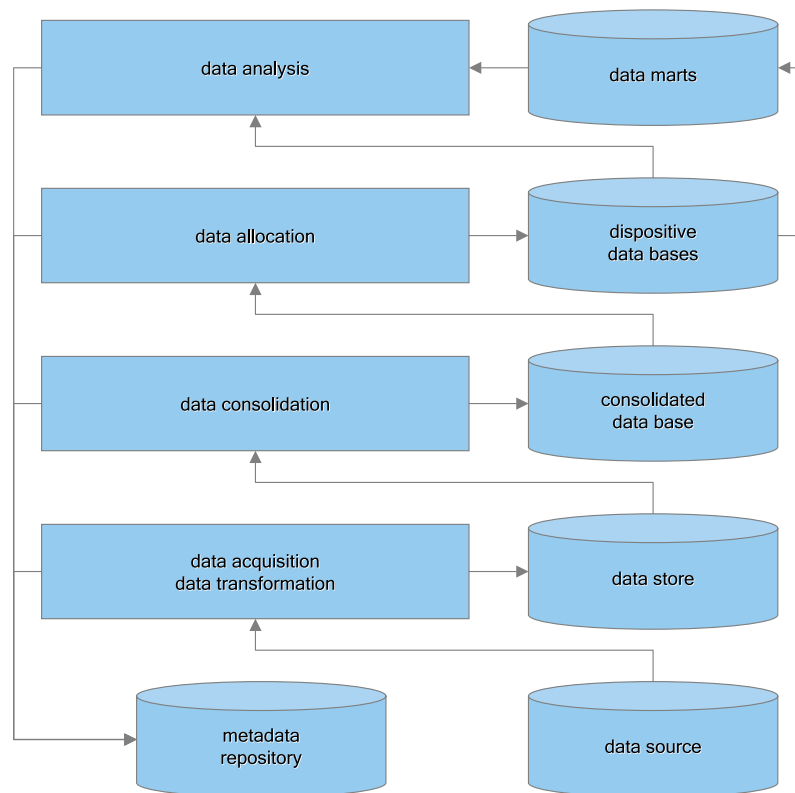


Figure 2.1: Components of a Data Warehouse System (Lehner, 2003)

- *Data sources:* Systems or files that are direct or indirect suppliers of data for the data warehouse.
- *Component for data acquisition and transformation:* A data pool which integrates data that has been extracted from the data source. Raw data is loaded into this component (*load time*) first, and then it is cleansed and transformed in order to achieve data integrity in the data warehouse.
- *Component for data consolidation:* After the cleansing process, data is added to the consolidated data basis of the data warehouse (*refresh time*). The consolidated data basis is an organization wide, application-independent storage of detail data for the data warehouse.
- *Component for data allocation:* Contrary to the consolidated data basis, the dispositive data basis is biased toward a specific application. Data in a dispositive data basis is structured or summarized according to the requirements of the application. New data in the consolidated data basis is updated in the dispositive data basis.
- *Component for data analysis:* This component encompasses all applications and data bases that serve as specific user interfaces for exploration of the data material. Typically, a multidimensional

data mart, in the sense of (Inmon, 1996), is a multidimensional database specifically designed for multidimensional data analysis, which is fed by a dispositive data basis.

- *Metadata repository:* All processes and structures within a data warehouse system should be documented. The metadata repository component serves as a storage system for metadata, which describes the data sources of a data element as well as the applied processes and transformations.

To summarize, a data warehouse system is a collection of databases and processes that allows collection, integration, storage, allocation and analysis of data. Furthermore, metadata stores for each data element the path and transformations it experienced during each process.

## 2.3 OLAP

According to (Lehner, 2003), the primary aim of a data warehouse system is to integrate data from a large number of sources to provide a consolidated data basis, which serves as a data basis for several analysis applications. There are many different analysis tools, for example standard reporting, data mining, or data visualization. The most well known application, which is often mentioned in the context of data warehouse, is OnLine Analytical Processing, or OLAP.

### 2.3.1 Definition of OLAP

The authors of (Codd, 1993) describe an approach to database processing which they call OnLine Analytical Processing (OLAP), being opposed to OnLine Transactional Processing (OLTP). The authors argue that the data volume of organizations is constantly growing, and that the proper use of this data is a competitive advantage. They describe the major shortcoming of many DBMS products: “Most notably lacking has been the ability to consolidate, view, and analyze data according to multiple dimensions.”

In traditional database front-ends, once the data had been retrieved from the database, the ability to aggregate, sum, consolidate or analyze that data was extremely limited. The solution the authors propose is OLAP, a framework for multidimensional data analysis. The concept of an OLAP server is quite similar to the one of a data warehouse system: it plays a mediating role for heterogeneous information systems. Multidimensional analysis is an application which runs on the data provided by the OLAP Server. This kind of analysis allows the navigation and manipulation of data using many consolidation paths called *dimensions*.

The authors have created a list of features that a product should have in order to be suitable for OLAP. The following summarizes the most important points of that list:

- *Multidimensional conceptual view:* The user view of data should be multidimensional, because it facilitates data analysis. Slicing, dicing, pivoting and rotation of consolidation paths (dimensions) should be possible.
- *Transparency:* Facts such as data residing on a remote server, or that data being retrieved from a heterogeneous data source, should be concealed from the user.
- *Accessibility:* The analysis tool must map its own logical schema to heterogeneous physical data stores, access the data, and present a single, coherent and consistent user view.
- *Consistent reporting performance:* As the number of dimensions or the size of the data base increases, the user should not perceive any significant degradation in reporting performance.

- *Client-server architecture:* It is mandatory that the OLAP product operates in a client-server environment, since most of the users work with personal computers, and most of the data is stored on remote database servers.
- *Generic dimensionality:* The basic data structure, formulas, and reporting formats should not be biased toward any one data dimension.
- *Multi-user support:* The OLAP tool should provide concurrent access to many users at the same time.
- *Intuitive data manipulation:* Manipulation of consolidation paths, such as slicing, dicing and roll-up, should be possible by direct action upon the cells, without menu access.
- *Unlimited dimensions and aggregation levels:* The model should not restrict the user analyst to a certain number of dimensions. In practice, the authors suggest fifteen to twenty possible dimensions.

Unfortunately, (Codd, 1993) is not an independent research article, but a white paper sponsored by Hyperion Solutions. It is no surprise that the authors conclude their paper showing how Hyperion Essbase satisfies all the previously mentioned conditions for OLAP products. Anyway, the conclusion the authors give is itself a motivation for OLAP: “The quality of strategic business decisions made as a result of OLAP is significantly higher and more timely than those made traditionally.”

### 2.3.2 Multidimensional Data Analysis

Data analysis aims at extracting information from data. Multidimensional data analysis is the process of consolidating data with respect to certain data domains called *dimensions*. These dimensions structure the data space of analysis, which allows to calculate a grid of specific statements for every combination of dimension values.

Multidimensional information is often described as a *data cube*. Dimensions present structural information, similar to coordinates in a three-dimensional cube. A data cell is addressed by a tuple of dimension values. Every data cell contains a *fact*. A fact is an aggregated factual information about a *measure attribute*. The measure attribute is the attribute whose values are aggregated according to the cube’s dimensions.

For example, consider a three-dimensional data cube as shown in Figure 2.2. There are three dimensions, called *age*, *city* and *work*. Let us assume that we want to express facts regarding the number of patients in a certain city, of a particular age and in a specific work branch. Thus, the measure attribute in our example cube is the attribute *patient*, and the dimensions are *location*, *age* and *work*.

The data cell addressed by the dimension values *age* = 26, *work* = IT and *city* = *Fribourg* contains a fact, which is an aggregation of a set of tuples satisfying these dimensional conditions. In this example, we aggregate the tuples using the *count* aggregation function. The fact of a data cell is calculated by counting the number of measure attribute values (patients) which satisfy the dimensional conditions of the cell.

According to (Codd, 1993), multiple dimensions are used for data analysis. “Data consolidation is the process of synthesizing pieces of information into single blocks of essential knowledge. The highest level in a data consolidation path is referred to as that data’s dimension.” *Data consolidation* means summarizing tuples according to dimension values (or categories). A consolidation path is a series of categories for the dimension values. If dimension attributes are categorized, their domain is partitioned by values in the same category. This kind of hierarchical categorization leads to a consolidation path with different levels of categorization. For example, consider a dimension called *TimeOfVisit*. As

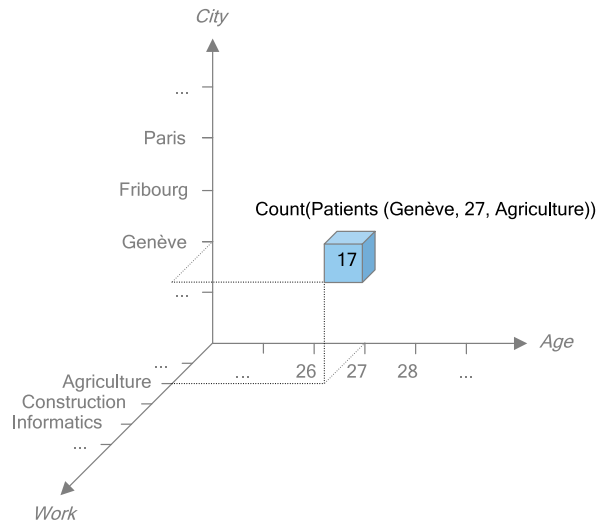


Figure 2.2: *Multidimensional Information on Patients*

illustrated in Figure 2.3, the domain of possible time stamps can be categorized by day, month or year. In our previous example, a consolidation path for the attribute `city` could be `location`, where `city` summarizes to `region`, and `region` summarizes to `country`.

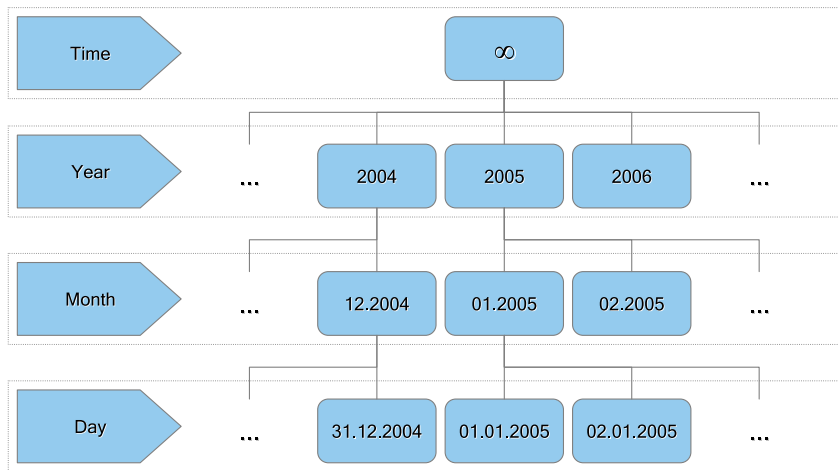


Figure 2.3: A Consolidation Path for Attribute `TimeOfVisit`

As described in this section, OLAP data analysis lets the user report facts in dependency of dimensions. While analysis is proposed to be multidimensional, the preferred physical storage scheme in (Codd, 1993) is *relational*. (Lehner, 2003) states that multidimensional data modeling on the level of the storage system, be it physical or logical, is central for data warehouses. Since data warehouses are built with an orientation toward analysis, he thinks that multidimensional modeling is the best approach. The author distinguishes qualifying information from quantifying information. Dimensions present qualitative information, while the facts are quantitative.

Anyway, there are reasons against multidimensional modeling at the core of the data warehouse system. Many researchers think that it is important to treat dimensions and measures symmetrically. With this approach, measure attributes and dimension attributes can be interchangeably defined within the process of analysis. For example, two different cubes can be built on the same data space, where in the first cube, an attribute is a dimension, and in a second cube, this same attribute functions as measure. Perhaps it is best to leave dimensional modeling to an OLAP application, especially the definition of quantitative and qualitative attributes.

In fact, this is the approach that Bill Inmon had in mind. In (Inmon, 1996), he distinguishes between the data warehouse on one hand, and *data marts* on the other. The latter are multidimensional databases specially built for OLAP, which are fed with data from the integrated data basis of the data warehouse.

### 2.3.3 OLAP Operators

The manipulations of consolidation paths in multidimensional data analysis are often referred to as the application of OLAP operators to data cubes. There are several OLAP operators that are common to all OLAP products. The following points summarize the ones most frequently used, and gives examples referencing Figure 2.2.

- *Slice*: Slicing a cube by a value means fixing the value of one dimension. The resulting cube contains only the data cells which are addressed by the slice value. For example, if we slice the dimension `city` in Figure 2.2 by the value `Genève`, only data cells with a location equal to `Genève` will be contained in the resulting cube.
- *Dice*: Dicing a cube means restricting the domain of the cube's dimensions. The resulting cube contains data cells within the restricted domain of the dimensions. A dice operation corresponds to a slice operation on multiple dimensions using multiple slice values. For example, we could restrict the domain of the dimension `location` to locations in Switzerland. The resulting cube only contains data cells where the location is in Switzerland.
- *Drill*: To drill a cube dimension means to change the consolidation level of that dimension. Drill-down stands for changing the level toward more detail or finer granularity. Roll-up means to change the level toward more summarization or coarser granularity. For example, we could roll-up dimension `location` from `region` to `country`, or drill it down to `city`.
- *Pivot*: Pivoting a cube changes the order of dimensions. This is not really a cube operator, since it does not affect the data cells. Pivoting is a visualization feature of the user interface.

### 2.3.4 OLAP Implementation

There are different possible OLAP implementations. Whereas the user interface allows a multidimensional access to the data, this data does not necessarily have to be represented in a physically multidimensional data structure.

If the underlying data base for the OLAP application's data structure is relational, we speak of ROLAP, or Relational Online Analytical Processing. (Codd, 1993) proposes a relational data structure for data storage with a multidimensional *user interface*, but the underlying data remains relational.

For ROLAP applications, the usual data model is the star or snowflake schema. As described by (Meier, 2001) and (Colossi *et al.*, 2002), this schema implements a fact table, which contains the facts for every combination of dimension values, and a set of dimension tables containing the dimension values and their consolidation levels. As illustrated in Figure 2.4, the fact table contains values for

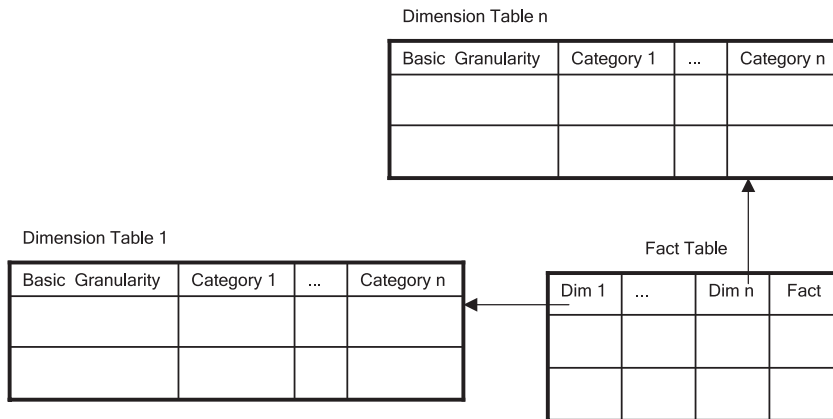


Figure 2.4: *The Star Scheme: Fact Tables and Dimension Tables*

combinations of dimension values of basic granularity. The dimension tables indicate to which category the dimension values belong. Based on the dimension tables, the fact values can be aggregated according to the consolidation levels using the corresponding dimension categories.

If the underlying data base has a physically multidimensional structure, that kind of implementation is called MOLAP, or Multidimensional Online Analytical Processing. (Lehner, 2003) points out that, although many have propagated the adequacy of physically multidimensional data storage, there has been little success of such systems. The author mentions multidimensional B\*-trees, a combination of one-dimensional index structures, as possible physical implementation.

## 2.4 Data Warehousing Technology

This section describes four technologies that have recently been or are still being developed for data warehousing and OLAP: First, extensions of the SQL language introduced in order to provide a certain OLAP functionality; second, MDX, a multidimensional query language; third, CWM, an interchange format for data warehouse metadata; and fourth, JOLAP, a Java API for OLAP.

### 2.4.1 SQL Extensions

(Lehner, 2003) and (Colossi *et al.*, 2002) describe some extensions of SQL for OLAP, which are supposed to add some OLAP functionality to relational databases. These additional SQL-operators, first introduced by (Gray *et al.*, 1997), are CUBE, ROLLUP and GROUPING SETS. In fact, these operators are extensions of the GROUP BY clause, which is mostly used in conjunction with aggregate functions. Those aggregate functions return a single value as a result of an operation conducted on a set of values. The GROUP BY clause groups the set in order to provide a series of sets for use with aggregate functions (Kriegel & Trukhnov, 2003). The following points summarize the explications of those operators in (Colossi *et al.*, 2002):

- **ROLLUP**: This operator is an extension of the GROUP BY clause which, based on a list of columns, generates multiple subtotal grouping clauses. This has the effect of a hierarchical calculation of sum values in a dimension of a cube.

- **CUBE**: This operator is an extension of the `GROUP BY` clause that generates subtotals for all permutations of the grouping columns, as well as the grand total. Applying the cube operator to a rollup for many dimensions returns a result that represents a hierarchical OLAP cube.
- **GROUPING SETS**: This operator allows multiple grouping clauses in a single statement. The primary benefit of this operator is that it limits the size of summary tables.

According to (Colossi *et al.*, 2002), in the early 1990, the SQL language had performance problems in implementing large OLAP cubes: `GROUP BY` could not return results for cells with different levels of aggregation; when aggregates for various levels were computed, rows of the fact table were read repeatedly; SQL supported only a few aggregation functions; metadata was not part of the database catalog, and thus, structural information was not available to query compilers. The operators described above provide a more efficient way for OLAP applications to access a relational database for summarized data.

### 2.4.2 MDX

Multidimensional expressions, or MDX for short, is a multidimensional query language developed for Microsoft OLAP Services (Messerschmidt & Schweinsberg, 2003). (Lehner, 2003) provides a short but expressive description of MDX. The syntax is very similar to SQL. A general MDX query has basically the following structure:

```
select
  <dimension values> on rows
  <dimension values> on columns
from <cube>
where <conditions>
```

The result of an MDX query is always a two-dimensional table. The `select` clause selects axis dimensions of the two-dimensional result space of the query. The keywords `on rows` and `on columns` have the effect that their corresponding dimension values are displayed in the row headers respectively in the column headers of the result table. The `from` clause indicates the data cube from which to choose the results. Finally, the `where` clause allows to restrict the domain of a cube by a set of predicates.

### 2.4.3 CWM

The problem of proprietary metadata formats is a restricted interoperability of heterogeneous systems, because the different metadata structures make the metadata exchange more difficult. The Open Management Group (OMG) has published a standard (OMG, 2003) for data warehouse metadata interchange in 2001, called the Common Warehouse Metamodel (CWM). A metamodel is a framework for the description of models. CWM gives guidelines for the design of data warehouse models. The aim of this standard is to provide a flexible, vendor-independent and formally defined framework for modeling structures and processes in data warehouses in an easily interchangeable way.

This specification is based on accepted standards such as UML (Unified Modeling Language) or XMI (XML Metadata Interchange). It provides a framework for the metadescription of management, analysis, resources, foundations and object models of data warehouses. Additionally, it allows custom extensions CWMX (CWM Extensions). The standard is quite extensive; the specification has about six hundred pages. CWM is a metadata modeling framework developed by and for industrial corporations. However, for the prototype which has been developed in this thesis, applying the CWM metadata model would present an unjustified effort. It would not make sense to comply to a heavyweight standard whose benefit is simplified metadata exchange, if the prototype does not intend to exchange metadata. That is why a simple E/R framework for metadata modeling has been used.

#### **2.4.4 JOLAP**

There is a Java specification request (JSR) named JOLAP (JSR, 2003) which is still under development. The specification already has the status of a final draft, which is now in the process of being approved by a committee. This committee consists of renowned members such as Apache Software Foundation, Apple Computer, Borland, Fujitsu, Hewlett-Packard, IBM, Macromedia, Nokia, Oracle, SAP or Sun Microsystems. The JOLAP specification aims at the development of a Java API for programming OLAP applications. A JOLAP-enabled database product will facilitate the development of dimensional reporting tools. The JOLAP API will be based on the same standards as CWM, as well as on CWM itself. Since JOLAP is still under development, it is definitely worth a periodical examination, since it promises to be the standard for the development of Java Applications for OLAP in the future. Since the JOLAP API package is not yet available, it has not been used for the implementation of the thesis prototype.

## Chapter 3

# E-Health and Healthcare Information Systems

This chapter begins with definitions of the e-Health concept. Then it shows some interesting fields of application, such as telemedicine, electronic patient records, the electronic health card, electronic prescriptions or mobile healthcare applications. After that, the use of data warehouses in healthcare is explained and a case study is described. Finally, ethical and legal issues of e-Health are stated.

### 3.1 Definition of E-Health

The term e-Health is generally understood as the use of information and communication technology (ICT) in healthcare. (Warda & Noelle, 2002) understands the term synonymously with health telematics, which means the use of telecommunication informatics in healthcare. (Dietzel, 2003a) defines e-Health as a description of all services, quality improvements and rationalizing effects which can be achieved through the digitalization of data and communication processes in healthcare. Contrary to (Warda & Noelle, 2002), (Jähn & Nagel, 2003) explicitly distinguishes this term from the notion of health telematics. In that understanding, the concept may contain *any* application of informatics in healthcare, such as telecommunication systems, information systems, or scientific computing. Every IT platform deployed in benefit of healthcare quality can be understood by this meaning of the term, for example computer aided disease diagnosis or electronic patient records.

The concept of e-Health is quite new, which is why the definitions are not yet standardized. It is thus arguable whether data warehousing in healthcare can be classified as e-Health applications. Depending on the definition of e-Health, if that definition encompasses medical information systems for decision support, then data warehousing can be part of an e-Health application.

### 3.2 Fields of Application for E-Health

In (Jähn & Nagel, 2003), the term e-Health is understood as “the variety of aspects which arise from the combination of information and communication technology with medicine-related fields of application - with a focus on the consequences. Especially in this context, the prefix “e-” is written in lowercase, and the term “Health” is written in uppercase.” This book presents an overview of the possibilities and consequences that arise from the combination of healthcare and ICT. An important project is the

electronic health card, which, together with the electronic patient record and the electronic prescription, will fundamentally change the healthcare system as we know it.

- As described by (Markus T.J.Mohr, 2003), the concept of telemedicine encompasses the electronic distant exchange of case related diagnostic and therapeutic data. The related communication process between medical professionals is called teleconsulting. For example, (Dietzel, 2003b) describes how teledermatology allows a dermatologist to transmit microscopic images of skin changes in order to get a diagnostic evaluation or a secondary opinion. The same process can be applied in any other field of application where digital images or descriptions of the patient's conditions can be transmitted. Telepathology, teleradiology or even telepsychiatry are mentioned. The unifying feature of these applications is that the emplacement of medical counseling or diagnosis is different from the location of the patient, and that distance is made possible by computer telecommunication.
- In Germany and Switzerland, there is a current development which will replace the old health insurance card: The electronic health card (e-healthcard), as explained by (Ramming, 2003a) and (Dietzel, 2003a). It consists of a microchip on a plastic card containing the ID number of the health insurance client. As illustrated by figure 3.1, the chip on the card will provide memory space for administrative data, for emergency data as well as for electronic prescriptions. Most importantly, there is memory for the electronic patient record.

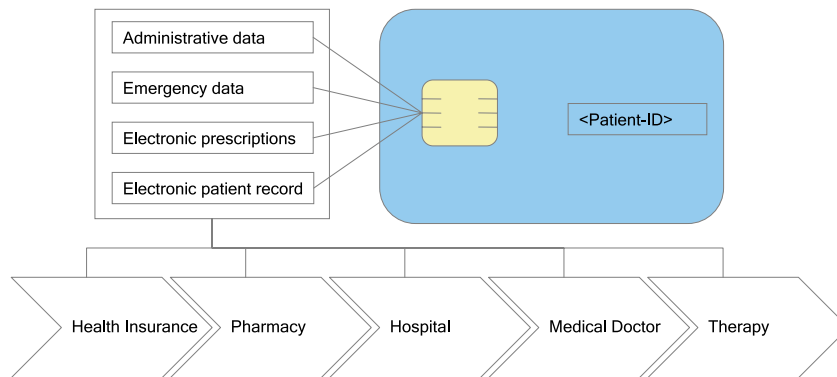


Figure 3.1: Role of the electronic health card in integrated healthcare

This card's purpose is to document the patients diagnoses, medical treatments, disease history and current health status in order to prevent double examinations. Furthermore, the card will play an important role concerning electronic prescriptions. In emergency situations, the information on the card may even save lives. The card will be accessed by pharmacies, medical doctors and hospitals as well as therapy centers. Electronic data transfer promises to be more efficient than paperwork. In Switzerland, a field test is currently running in Lugano (Borchers, 2004). Since November 2004, the Carta Sanitaria (<http://www.retesan.ch>) is being tested over a time period of 18 months. In Germany, field tests will start in 2005.

- Electronic drug prescription is one of the key applications of the electronic health card. Prescriptions will be digitally stored on the patient's chip card. This will improve the processing speed of the data in connection with prescriptions. Pharmacies can directly process prescriptions, and check them for unwanted side effects or individual incompatibilities of the prescribed drugs.

Processing will be faster, more efficient, and more accurate. Up to now, only prescription-free drugs can be sold over the internet. (Apermann, 2003) goes so far as to propose an online pharmacy, where electronic prescriptions on the electronic health card could enable a patient to order prescription drugs online.

- (Schramm-Wölk & Schug, 2003) distinguishes different levels of electronic patient records (EPR). On the first level, there is computer support for the creation of a medical record on paper. On the second level are scanned medical records using document imaging. Level three encompasses an exclusively computer-generated medical record for an institution. Level four denotes a patient-centered collection of data from medical records of level three from different institutions. Together with the introduction of electronic health cards, the electronic patient record of level four will arrive. That is, the complete medical history of a patient will be stored on his health card. On one hand, if that card is the only central storage, then the patient is responsible to whom he is willing to disclose information about his medical record. We will discuss the ethical and legal aspects of electronic patient records later.

The electronic patient record has an important role in the model of integrated health care presented by (Ramming, 2003b), as shown in figure 3.1. All stations in a series of treatments, beginning from the general practitioner, over the specialist and the hospital to rehabilitation therapy, access the EPR in order to retrieve necessary parts of the patient's medical record.

### 3.3 Data Warehousing and E-Health: The Clinical Data Warehouse

The motivation for the use of a data warehouse in healthcare can be described by its contribution to the healthcare value. This value can be modeled by a value function (Schilp & Gilbreath, 2000):

$$Value = \frac{Clinical\ Quality + Customer\ Service}{Cost}$$

That value indicates the relationship between quality and service on one hand and cost on the other. Considering that function, the motivation for a data warehouse is either to increase quality or customer service, or to decrease cost. For example, in a hospital, much time is lost because of inefficient storage of information. With an efficient data storage, hospital employees spend less time for clerical work, which leaves them more time for patient care.

The aim of a clinical data warehouse is to transform the electronic patient record into aggregated, multidimensional information. (Ledbetter & Morgan, 2001) propose a clinical data warehouse (CDW) that is fully integrated into the electronic patient record and the clinical decision support system. The CDW does not contain a mirror image of the data in the transactional system, but a subset of data useful for retrospective aggregatio analysis.

A data warehouse in healthcare supports the medical staff in their decisions. The system provides information upon which the decision makers can base their judgment. This information can be administrative in nature, it can concern the management of the hospital's infrastructure, or even the treatment of patients. Anyway, the decision makers will try to get useful new information through the retrospective analysis of available data.

The following subsection discusses a case study of a data warehouse implementation in an academic medical center.

### 3.3.1 Case Study: A Clinical Data Warehouse Implementation

At the university of Virginia, a clinical data warehouse called clinical data repository (CDR) has been implemented (Einbinder *et al.*, 2001). CDR presents a ROLAP implementation of a data warehouse, which draws data from four different information systems. First, there is the record of patient visits, available from shared medical systems. Second, billing transactions are imported from a billing information system. Third, laboratory results are integrated. And fourth, clinical details for thoracic surgery cases are stored inside the data warehouse. This system provides data integration on heterogeneous sources. The developers plan to integrate more sources, such as microbiology results or outpatient prescribing information.

The user interface runs in a standard web browser. SQL statements are generated automatically in response to point-and-click actions by the user. The interface enables ad-hoc queries without knowledge of SQL. The application generates SQL statements, passes those to CGI programs which execute database queries, and return dynamically generated web pages. Queries are not multidimensional, but relational in nature. A user can define a population of interest by setting conditions for the query. However, no explicit OLAP functionality is described, neither roll-up, slicing nor dicing.

The case described by (Einbinder *et al.*, 2001) presents an example of how information sources can be integrated in order to create a central data repository for a hospital or medical center. Yet, apparently the system does not provide more analytical OLAP-functionality. More specifically, what is lacking is multidimensional data analysis, that is the calculation of facts in dependency on several dimensions and their consolidation levels, together with operators to manipulate these facts.

## 3.4 Ethical Issues

The sixth paragraph of the Hippocratic Oath reads “Whatever in connection with my professional practice or not in connection with it I see or hear in the life of men which ought not to be spoken of abroad I will not divulge, as reckoning that all such should be kept secret.” When information is encoded digitally, every copy is an original, and it can be transmitted to everywhere within no time. The same holds for medical information. Thus, privacy of medical information in an e-Health application must be ensured - especially when it comes to the electronic patient record.

(Krohs, 2003) proposes the analogical application of biomedical ethics described by (Beauchamp & Childress, 2001) to the field of e-Health. The four basic principles are respect for autonomy, beneficence, non-maleficence, and justice. First, patient’s actions should not be subject to external influences. Second, all actions should be chosen in order to maximize the patient’s benefit. Third, actions should not harm the patient in any unnecessary way. And fourth, the actions should be just in a way that treats all patients equally.

If we apply these principles to e-health, then this means the use of information- and communication technology with respect for the autonomy of the patient, in a way that is beneficent and non-maleficent, and just. If a computer system used for healthcare complies with these principles, then that system can be said to be an ethical e-Health application.

## 3.5 Legal Issues

There are several legal questions that arise with the use of ICT for healthcare (Schlegel, 2003). First of all, telecommunications rise the question of jurisdiction. The principle of territoriality states that all activity principally falls under the jurisdiction of the emplacement of that activity. Another issue is the regulation of information that may or may not be published. For example, in the European Union one

is not allowed to advertise in public for a prescription drug. Moreover, the electronic patient record concerns legal issue of data protection. The patient has the right to know what kind of information is stored in his file, and he has the right to refuse disclosure of information. In Switzerland, the current legislation on data protection is enough restrictive to be applied to the sensitive data which is processed in e-Health applications. (Schlegel, 2003) points out that their discussion mentions only a small part of the legal issues that will arise with the pervasion of e-Health technology. Furthermore, a legal judgment of the issues is still far from clear, and will only emerge after the spread of e-Health technology.

## **Part II**

# **Theoretical Background**

## Chapter 4

# Epistemology of Information Systems

Epistemology (Moser *et al.*, 1993) is the philosophical study of the nature, sources, and limits of knowledge. In this chapter, the term epistemology of information systems is understood analogically as the study of the nature of knowledge in relation to its representation and processing in information systems. The motivation for this chapter is to introduce basic formalisms of knowledge representation used in information systems, such as propositions, predicates, relations, or dimensions, in order to make use of these notational systems later in this thesis. Additionally, this philosophical chapter title allows a deeper philosophical examination of knowledge, information and data in connection with strategic information systems. First, the notions of knowledge, information, and data are analyzed. Then, the concepts of propositional, relational and multidimensional data is discussed in detail.

### 4.1 Knowledge

In (Eysenck & Keane, 2000), knowledge is defined in terms of *propositions*, which represent the ideational content of the mind. Propositions are knowledge atoms, smallest statements about objects and the relationship between them, which cannot be further subdivided. Usually, these propositions are represented using predicate calculus. For example *on(book, table)* states the spatial relationship between two objects of the world.

A somewhat different definition we find in (Davis, 1990): Knowledge is described as the state of a machine, which is said to know proposition  $P$  when it is in state  $S$ , if, whenever the machine is in state  $S$ ,  $P$  is true. This is quite an abstract definition, but interestingly, knowledge is not restricted to human beings. Basically, knowing something means, for an abstract machine, being sure that the logical implication  $S \rightarrow P$  holds. A definition of knowledge which was generally accepted until the Gettier problem was suggested by Plato (Meno, 97e -98a): *Knowledge is justified true belief*. That is, knowledge consists of propositions which we believe to be true, that we have some justification for believing so, and that are actually true.

Aristotle wrote about the generally knowable in *Metaphysics*, Book III, Part 4. “*All things that we come to know, we come to know in so far as they have some unity and identity, and in so far as some attribute belongs to them universally.*” Knowable substance implies *unity, identity* and *attributes*. Interestingly, relational tuples present building blocks of knowable substance in the sense of Aristotle: Every distinguishable tuple is a unity in form of a collection of values. The tuple identity is a symbol that identifies each tuple, which is given by the primary key attribute. And finally, the attributes of a tuple are the instances of attribute values for every attribute of the relation.

Philosophers do not agree on what knowledge is. Furthermore, they still discuss whether or not it is possible at all to know anything for certain. This is called skepticism. It has a lot to do with the concept of truth. Therefore, it is courageous to talk about knowledge as quantifiable commodity, as do certain adepts of knowledge management. However, philosophers seem to presuppose a concept which they call *belief*: A proposition is a representation of a state of the world that is possibly true. Our *belief* is the set of propositions which we believe to be true. The epistemological question of how we can *really know* for certain that such a belief is true is not interesting in connection with information systems. If we concentrate on the connection of knowledge and information systems, we must ask “what is a proposition, and how can it be represented”? The *knowable* (in the sense of Aristotle) is the stuff that is described by every data model, and referred to by any data base entry. Therefore, the following sections will concentrate on representations of the knowable. A *Logic* is a schema for the development of languages for knowledge representation. In the following subsections, we will present the propositional calculus as well as the predicate calculus, which are frequently used logical systems for knowledge representation.

### 4.1.1 Propositional Calculus

Classical, two valued propositional calculus is the discipline of reasoning about propositions. A proposition can be any statement to which a truth value (true or false) can be assigned. In propositional calculus, those propositions are represented by variables. According to (Nguyen & Walker, 1997), its syntax is defined by a set of propositions  $V$ , a set of formulas  $F$  and a truth function  $\tilde{t}$ .

Let the set  $V = \{v_1, v_2, \dots\}$  be the set of elementary propositions (statements, variables), which can be true or false. The set  $F = \{\phi_1, \phi_2, \dots\}$  of formulas can be constructed by composing propositions using the Boolean operators  $\wedge$  (and),  $\vee$  (or) and  $\neg$  (not): Every proposition is a formula. If  $\varphi$  and  $\psi$  are formulas, then  $\varphi \wedge \psi$ ,  $\varphi \vee \psi$  and  $\neg\varphi$  are formulas.

The semantics of propositions is described by a truth function  $t : V \rightarrow \{0, 1\}$ . This function is a mapping from propositions to truth values, which assigns to every proposition a value 1 (true) or 0 (false). This truth function can be extended to formulas. The function  $\tilde{t} : F \rightarrow \{0, 1\}$  is defined recursively over the semantics of the Boolean operators:

- If  $p$  is a proposition,  $\tilde{t}(p) = t(p)$ .
- If  $\varphi$  is a formula,  $\tilde{t}(\neg\varphi) = \begin{cases} 1 & \text{if } \tilde{t}(\varphi) = 0 \\ 0 & \text{else.} \end{cases}$
- If  $\varphi$  and  $\psi$  are formulas,  $\tilde{t}(\varphi \wedge \psi) = \begin{cases} 1 & \text{if } \tilde{t}(\varphi) = \tilde{t}(\psi) = 1 \\ 0 & \text{else.} \end{cases}$
- If  $\varphi$  and  $\psi$  are formulas,  $\tilde{t}(\varphi \vee \psi) = \begin{cases} 0 & \text{if } \tilde{t}(\varphi) = \tilde{t}(\psi) = 0 \\ 1 & \text{else.} \end{cases}$

### 4.1.2 Predicate Calculus

As explained in (Davis, 1990), the predicate calculus, also known as first order logic, is the most important and commonly used logical system. It is known to be sufficiently powerful for classical mathematics. The predicate calculus presents an extension of propositional calculus by giving propositions an inner structure. This subsection will explain some basic concepts, but we are not going into details. The point is to show the connection between relations and the semantics of predicate logic. Thus, we are omitting the notion of quantification here.

A language  $L$  described in predicate calculus contains function symbols and predicate symbols. Every function symbol with zero places (a *constant symbol*) is a *term*. If each  $t_i$  is a term, then  $f(t_1, \dots, t_k)$  is a term, where  $f$  is a  $k$ -place *function symbol*. If each  $t_i$  is a term, then  $p(t_1, \dots, t_n)$  is a *formula*, where  $p$  is a  $n$ -place *predicate symbol*. Furthermore, connections of formulae with Boolean operators, as described in the previous section, are formulae.

As illustrated by Figure 4.1, the semantics of a *language* in predicate calculus is defined by an *interpretation* of the *function and predicate symbols* in a *universe of discourse*. The semantics of a predicate calculus language  $L$  is derived from an interpretation  $I$  within a universe of discourse or domain  $D$ . This kind of semantics is called Tarskian Semantics:

- An interpretation  $I$  associates
  - every constant symbol  $c$  in  $L$  with an *element*  $c^I \in D$ ;
  - every  $k$ -place function symbol  $f$  with a *function*  $f^I : D^k \mapsto D$ ;
  - and every  $n$ -place predicate symbol  $p$  with a *relation*  $p^I \subseteq D^n$ .
- If in an interpretation  $I$ , a term  $t$  stands for an element of the domain  $d \in D$ , we write  $t^I = d$ . The truth value of a formula  $\varphi$  in an interpretation  $I$  is denoted  $\varphi^I$ .
- A predicate statement  $\varphi = p(t_1, \dots, t_n)$  of  $L$  is true if the corresponding relation  $p^I$  contains the tuple  $(t_1^I, \dots, t_n^I)$ . That is,

$$(t_1^I, \dots, t_n^I) \in p^I \Rightarrow \varphi^I = 1$$

$$(t_1^I, \dots, t_n^I) \notin p^I \Rightarrow \varphi^I = 0$$

- Connections of formulae with Boolean operators have the same semantics as in propositional calculus.

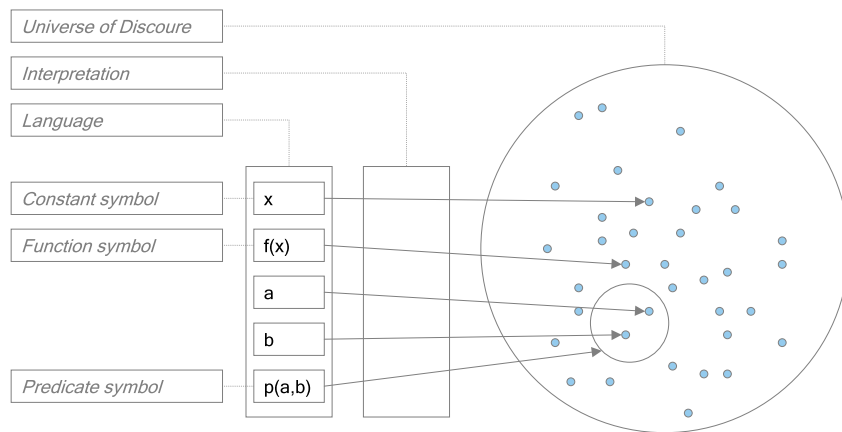


Figure 4.1: *Semantics of the Predicate Calculus*

Predicate calculus allows to form languages powerful enough to describe mathematics. The semantics of those languages are determined by functions and relations. Since all functions *are* a special case of relations, we can express the semantics of a language entirely by a set of relations. For the epistemology of information systems, this means that *relations are powerful enough to represent any knowable substance* describable by a first-order language. The consequence for information system design is that relational database systems are sufficiently powerful for knowledge representation.

## 4.2 Information

Information theory is the study of entropy in signal transmission. In this context, information is defined as the change of uncertainty about a proposition. Signal transmissions change the probability for a given proposition to be true, and this change is the quantity of information, which is measured in bits. In this section, however, information quantity is neglected, and thus we are not describing the concept of information entropy in detail. What we are interested in is the *nature* of information insofar as it is used to support decisions in business processes. Thus, the question is, *what is the substance of information?*

(Claus & Schwill, 2001) distinguishes signals, messages, data, and information very clearly. On one hand, signals are elementary noticeable changes, data are signals, represented by digital symbols, and a message is a sequence of signals including its temporal and spatial arrangement. Information, on the other hand, is defined in terms of three parts: *syntax, semantics and pragmatics*. Syntax states how data has to be arranged, semantics denotes the meaning assigned to the data, and pragmatics means the intent of the message. What logicians often neglect is pragmatics. It can be observed that an information flow always implies a change in the state of the world. Thus, information is often emitted or collected with a specific intention.

As we have seen in the previous section, propositions can be expressed by a language  $L$ , where the elements of that language, called symbols, denote elements of a universe of discourse (function symbols) or the relationship between them (predicate symbols). If we consider the individuals of the domain as the objects of knowledge, then information is *a representation of those individuals or the relationships between them*. The syntax of information can be expressed by a predicate language  $L$ , while the semantics of information are given by an interpretation  $I$ .

Knowledge is in the mind, but information is an external representation. Knowledge is the intention, and information is the extension. Information is used to store or transmit propositional knowledge about the world. Information systems have been developed and built exactly for this purpose, but all they can handle are the symbols or data elements. For short, information is knowledge representation using data symbols.

In that light, information which is used for decision support ought to provide a manager with *propositions* (that is, knowledge) which imply the right (or most valuable) decision. A data warehouse must therefore allow the *consolidation* of huge amounts of data during the process of data analysis, in order to extract propositions that are true for a wide range of data elements. This consolidation is achieved by reducing many data elements to classes of data elements. The resulting statements are consolidated insofar as there are less statements for only a few element classes.

## 4.3 Data

(Claus & Schwill, 2001), contains a definition of data: A datum is a smallest, undividable element of a data type. In informatics, this term is often defined as a signal or a message which is encoded in a way that it is processable by a machine. In a nutshell, *data are signals that encode information, and information represents knowledge*. If we consider data as digital symbols, then information is a representation of knowledge consisting of those symbols.

Thus, data are symbols  $s$  from a *data domain*  $D$ . If these symbols  $s \in D$  are combined according to the syntax of a language, they can be used to convey meaning. For example, in written language, the basic domain of data symbols is  $\mathbb{D}^* \cup \mathbb{S}^*$ , where  $\mathbb{D} = \{0, 1, \dots, 9\}$  is the set of digits,  $\mathbb{S} = \{a, b, \dots, Y, Z\}$  is the set of letters, and  $S^*$  denotes the set of possible concatenations of elements from a set  $S$ .

In information systems, especially in a data warehouse, the relational and the multidimensional data model are basic models for data description. The following subsections explain these two models in

detail.

### 4.3.1 Relational Data Model

The relational database scheme was first published in (Codd, 1970) and is extensively explained in (Pirrotte, 1982) and (Meier, 2001). It is based on the mathematical concept of relations.

**Definition 1** A mathematical relation  $R$  over  $n$  sets  $S_1, S_2, \dots, S_n$  is a subset of the Cartesian product

$$R \subset S_1 \times S_2 \times \dots \times S_n$$

where each member  $t_i$  of this relation is an  $n$ -tuple  $(t_{i1}, \dots, t_{in})$ ,  $t_{ij} \in S_j$ ,  $j = 1, \dots, n$ .

If a relation is a subset of the Cartesian product of  $n$  sets, and contains  $m$  tuples, it can be visualized as a table of  $n$  columns and  $m$  rows.

**Definition 2** A relational attribute  $A_i$  in a relation  $R$  is a unique name for a set in the domain of  $R$ .

The set of possible values for an attribute  $A_i$  is called the domain of  $A_i$  and it is written  $dom(A_i)$ . The Cartesian product of the attributes' domains is called the domain of  $R$ . The set of attributes will be noted with a capital letter  $A$ . The value of an attribute  $A_i$  in a tuple  $t_j$  will be noted

$$A_i(t_j) := t_{ij}$$

Defined in terms of attributes, a relation is a subset of the Cartesian product over the domains of its attributes:

$$R \subset dom(A_1) \times dom(A_2) \times \dots \times dom(A_n)$$

**Example 1** Consider the relation 'patient', as shown in Table 4.1. This relation is a subset

$$patient \subset \# \times patient\_id \times bmi \times cstrl \times d\_bp \times age.$$

This relation is a simplified example of how a medical record in e-Health could look like. It stores for every patient with a patient-identification ('patient\_id') his or her body mass index (a key figure for measuring the healthiness of the body wheight) in the attribute 'bmi', the cholesterol level in 'cstrl', the diastolic blood pressure in 'd\_bp' and the patient's age in the attribute 'age'. The values are randomly selected.

This example relation will be used throughout this thesis to explain the notion of fuzzy context classification in OLAP-cubes.

In a database, tuples have an identification key, that is a combination of attributes which uniquely identifies each tuple, and which is minimal. Usually, this attribute is a numerical tuple identification.

In terms of predicate calculus, a relational scheme presents a first order language, where the tuple data elements are constant symbols, every attribute is a function symbol, and every relation is a predicate symbol.

The relational algebra consists of a set of operators that can be applied to relations. First, all set operators can be applied, since relations are themselves sets, namely sets of tuples. Thus, union  $\cup$ , intersection  $\cap$ , set difference  $\setminus$  and the Cartesian product  $\times$  are part of the relational algebra (Meier, 2001). Additionally, there are three important relational operators: selection  $\sigma$ , projection  $\pi$  and join  $\bowtie$ .

A selection on a relation  $R$  by a predicate  $P$  returns a new relation, which contains all tuples in  $R$  that satisfy  $P$ :

$$\sigma_P(R) = \{ t \in R \mid P(t) \}$$

Table 4.1: Relation 'patient', used as example throughout the paper

#	patient_id	bmi	cstrl	d_bp	age
1	A	19.7	140.7	92.4	35
2	B	19.7	172.5	78.2	65
3	A	19.3	125.4	73.8	36
4	B	19.9	129.1	74	66
5	E	28.7	167.5	115.2	26
6	F	27.1	283.7	96.5	64
7	G	17.2	142.2	91.8	45
8	H	19.25	160.3	75.4	34
9	I	15.8	233.6	94.6	76
10	J	22.4	167.2	107.8	72
11	K	21.6	228.3	84.5	48
12	L	24.2	256.9	68.6	94
13	M	20.9	144.5	93.7	59
14	N	22.5	187.1	85.4	56
15	O	18.4	238.9	87.8	34
16	P	23.2	174.7	86.7	56
17	Q	15.8	172.8	102.3	67
18	R	27.6	227.4	73.6	34
19	S	18.5	245.2	71.9	78
20	T	18.7	244.7	73.4	45
21	U	15.3	278.9	107.6	23
22	V	28.6	283.4	78.7	87
23	W	29.5	288.7	99.8	58
24	X	25.9	189.9	87.4	91
25	Y	26	190	87.5	19

A projection of a relation  $R$  on a set of attributes  $A_1, \dots, A_n$  returns a new relation which contains all tuples of  $R$  restricted to the projection attributes:

$$\pi_{A_1, \dots, A_n}(R) = \{ (A_1(t), \dots, A_n(t)) \mid t \in R \}$$

A join of two relations  $R$  and  $S$  is in fact equivalent to a selection on the Cartesian product of the two relations:

$$R \bowtie_P S = \sigma_P(R \times S)$$

### 4.3.2 Multidimensional Data

We already have seen in Section 2.3.2, in data analysis certain attributes can serve as dimensions of interest, while statements are calculated in dependency of dimension values. The Cartesian product of the dimension domains present a grid for the aggregation of a measure attribute with respect to the dimension values. This grid is often referred to as a *cube*. The measure attribute aggregation is called a *fact*. Every tuple in the Cartesian product of the dimensional attributes represents a *data cell* in that cube. Furthermore, for the purpose of data consolidation, dimensional values are assigned to classes which are called *dimension categories*.

In the multidimensional model, qualitative and quantitative attributes are distinguished (Lehner, 2003). Dimensions qualify, while measures quantify information. That distinction can be made on the level of multidimensional modeling. However, other approaches are possible, where all data domains are equally treated as dimensions, but only in a specific query (or cube), measures are chosen, and facts are calculated about the measure attribute value for the tuples in each data cell.

Many multidimensional data models have been developed (Vassiliadis & Sellis, 1999). There is no consent whatsoever on what multidimensional information is, or how it can be described. The following subsection will discuss two different models as examples: Lehner's model (Lehner, 2003), which distinguishes between dimensions and facts on the level of modeling, and a model developed by IBM Almaden Research Center (Agrawal *et al.*, 1997), which treats dimensions and measures symmetrically.

### Lehner's Model

The model described by (Lehner, 2003) incorporates a mechanism for dimensional data consolidation. It distinguishes on the level of data modeling between dimensions, facts and measures.

A *dimension scheme*  $D$  consists of a partially ordered set of category attributes

$$(\{ D_1, \dots, D_n, Top_D \} \rightarrow) \quad (4.1)$$

where  $\rightarrow$  denotes functional dependency and  $Top_D$  is a generic maximal element in relation to  $\rightarrow$ , such that  $Top_D$  is functionally dependent on all other category attributes. Furthermore, there exists exactly one  $D_i$  which functionally defines all other category attributes, and thus presents the finest granularity of a dimension.

In a multidimensional scheme, a *fact*  $F$  consists of a granularity  $G$  such that

$$F = G = \{ G_1, \dots, G_n \}$$

is a subset of all category attributes in the existing dimension schemes  $D_1, \dots, D_n$ . This subset satisfies the following conditions: (1) every  $G_i \in G$  is an element of a dimension scheme, and (2) every  $G_i \in G$  is functionally independent from any other  $G_j \in G$ .

Based on a fact  $F$ , a *measure*  $M$  is defined by a granularity  $G$  and a calculation rule  $f()$  over a non-empty subset of all existing facts

$$M = (G, f(F_1, \dots, F_k)).$$

A calculation rule is either a scalar function or an aggregation function.

Most importantly, a cube scheme  $C$  consists of a set of dimensional schemes  $D$  and a set of measures  $M$ :

$$C = (D, M) = (\{ D_1, \dots, D_n \}, \{ M_1, \dots, M_k \}).$$

An extension of a data cube is defined by a *Cartesian product* of the domains of all attributes contributing to the cube scheme.

The explicit dimensional categorization in this model is based on functional dependency. We will see that this can be equivalently expressed using contexts and equivalence classes. However, Lehner did not formally define the OLAP operators within his model.

In this thesis, multidimensional data modeling will be defined separately from data classification. That approach allows to simplify the multidimensional data model. However, dimension hierarchies will be introduced later by means of context classification. Furthermore, we will distinguish between the measure attribute, which is the attribute whose values are aggregated with respect to the dimension values, and the fact, which is the resulting value of the aggregation.

### A Symmetrical Multidimensional Data Model

That model presented by (Agrawal *et al.*, 1997) has been developed with a focus on implementation. The authors write that they wanted to stay as close to the relational algebra as possible in order to make operators translatable to SQL. Dimensional hierarchies are not treated explicitly in the model. Dimensions and measure attributes are treated symmetrically.

A cube  $C$  has the following components:

- $k$  dimensions each having a name  $D_i$  and a domain  $dom_i$ ;
- Elements defined as a mapping  $E(C)$  from  $dom_1 \times \dots \times dom_k$  to 0, 1 or an  $n$ -tuple.
- An  $n$ -tuple of names that describes the  $n$ -tuple element of the cube.

The elements of a cube can be either 0, 1, or an  $n$ -tuple.  $E(C)(d_1, \dots, d_n)$  refers to the element at “position”  $d_1, \dots, d_n$  of cube  $C$ . If the element corresponding to is 0 then that combination of dimension values does not exist in the database. A 1 indicates the existence of that particular combination. Finally, an  $n$ -tuple indicates that additional information is available for that combination of dimension values.

The authors continue by defining two operators, *push* and *pull*, which they regard necessary in order to treat measures and dimensions symmetrically. The *push* operation is used to convert dimensions into elements that can be manipulated using an aggregation function. Pushing dimension values to tuple elements append the dimension value to an  $n$ -tuple element. The *pull* operation creates a new dimension for a specified member of an  $n$ -tuple element. Thus, dimensions and measures can be transformed into each other.

The idea of that model is to allow all attributes to be dimensions, where measures are simply specially treated kinds of dimensions. The idea that will be followed in this thesis is that every data element in a multidimensional scheme is a tuple. There will be no distinction between elements that return 1 or 0, and elements that are tuples. That will facilitate the underlying data model as well as the symmetrical treatment of measures and dimensions.

## Chapter 5

# Fuzzy Set Theory and its Applications

In real world problems, we are usually dealing with uncertainty. Fuzziness is the kind of uncertainty arising from linguistic concepts without clear borders.

In natural language, many statements are vague, or fuzzy. For example, classifying a certain object as 'large' leaves us with an uncertainty of how large this object really is. Fuzzy concepts cannot be modeled by a simple set inclusion operator  $\in$ , but there is a degree of membership. Mathematically, fuzzy concepts are modeled by fuzzy sets, a generalization of crisp sets: "The modeling of fuzzy concepts by fuzzy sets leads to the possibility of giving mathematical meaning to natural language statements" (Nguyen & Walker, 1997). Thus, the motivation of fuzziness is *ergonomic knowledge representation*.

Fuzziness in software applications is useful for making user interactions more natural. For example, fuzziness in databases can lead to more intuitive and simpler queries. As another example, classification in data mining can be fuzzified. Often, classification features are not crisp, and that is where a fuzzy membership function can help out.

This chapter will explain the basic notion of fuzzy sets and will discuss fuzzy set operators; it introduces linguistic variables, fuzzy logic and approximate reasoning; and it shows two possible applications of fuzzy sets to data warehouses.

### 5.1 Fuzzy Sets

The concept of fuzzy sets has been introduced by (Zadeh, 1965). An ordinary subset  $A$  of a set  $X$  is determined by its indicator function, or characteristic function  $\chi_A$  defined by

$$\chi_A = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

The indicator function  $\chi_A$  of a subset  $A$  of a set  $U$  just specifies whether or not an element is in  $A$ . If we generalize the notion of characteristic function, we allow element  $x \in X$  to belong to a subset  $A$  of  $X$  to a certain degree  $\mu_A(x) \in [0, 1]$ . Accordingly, a *fuzzy subset*  $A$  of a set  $X$  is defined by a membership function  $\mu_A : X \rightarrow [0, 1]$ . For a fuzzy set  $A$ , the value  $\mu_A(x)$  is called the *degree of membership* of  $x$  in  $A$  (Nguyen & Walker, 1997).

The notion of power set is a concept of set theory. The power set of a set  $X$ , denoted  $P(X)$ , is the set of all subsets of  $X$ . Analogically, (Bandemer & Gottwald, 1993) denotes the set of all fuzzy subsets of a set  $X$  as  $F(X)$ . Let us call that notion the fuzzy power set of  $X$ .

## 5.2 Operators on Fuzzy Sets

In order to provide a complete fuzzy set algebra, operators on sets such as union and intersection must be fuzzified. The membership degree for the complement set of a fuzzy subset of  $X$  is usually calculated by  $\mu_{\overline{A}}(x) = 1 - \mu_A(x)$  for every element  $x \in X$ . However, there have been many implementations of fuzzy counterparts of union and intersection. In the following sections, the classical and the algebraic connectors are presented, as well as a compensating operator introduced by (Zimmermann, 1993).

### Classical Fuzzy Set Operators

When (Zadeh, 1965) introduced the notion of fuzzy sets, he intended the use of the minimum operator as fuzzy intersection, and the maximum operator as fuzzy union. Thus, using the operator *min* for the intersection of fuzzy sets, the membership degree of an element  $x$  to the intersection of two fuzzy sets  $A$  and  $B$  is the minimum of the membership degrees to the two sets:

$$\mu_{A \cap B}(x) := \min\{\mu_A(x), \mu_B(x)\}$$

The minimum operator is illustrated in Figure 5.1 using the surface graph of the function  $z = \min\{x, y\}$  for  $x, y \in [0, 1]$ , which yields a pyramid-like shape.

Analogically, if we use the operator *max* for the union of fuzzy sets, the membership degree of an element  $x$  to the union of two fuzzy sets  $A$  and  $B$  is the maximum of the membership degrees in each set. The maximum operator is visualized in Figure 5.2.

$$\mu_{A \cup B}(x) := \max\{\mu_A(x), \mu_B(x)\}$$

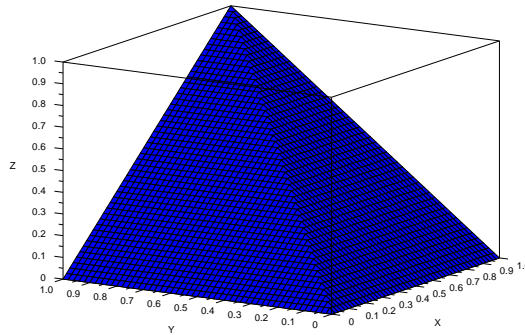


Figure 5.1: *Minimum Intersection*

### 5.2.1 Algebraic Fuzzy Set Operators

The algebraic product can be used to model fuzzy set intersection. Using the algebraic product, the membership degree of an element  $x$  to the intersection of two fuzzy sets  $A$  and  $B$  is simply defined as the multiplication of the membership degrees:

$$\mu_{A \cap B}(x) := \mu_A(x) \cdot \mu_B(x)$$

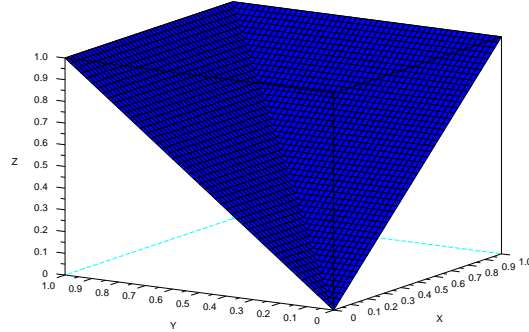
Figure 5.2: *Maximum Union*

Figure 5.3 shows a three-dimensional plot of the algebraic product in the interval from zero to one as illustration. In comparison to the graph shown in Figure 5.1, there is no edge of points where  $x = y$ , thus the transition is smoother. Furthermore, the truth value  $z = xy$  decreases faster with decreasing  $x$  and  $y$ .

Clearly, this operator is symmetric. The degree of membership of an element  $x$  to the intersection of  $n$  fuzzy sets  $A_i$  is defined by the product of the individual membership degrees:

$$\mu_{\bigcap_{i=1}^n A_i}(x) := \prod_{i=1}^n \mu_{A_i}(x) \quad (5.1)$$

The algebraic sum is the inverse function of the algebraic product, such that  $A \cup B = \overline{\overline{A} \cap \overline{B}}$ . The degree of membership of an element  $x$  to the union of two sets  $A$  and  $B$  defined by algebraic sum is calculated as follows:

$$\begin{aligned} \mu_{A \cup B}(x) &= \mu_{\overline{\overline{A} \cap \overline{B}}}(x) \\ &:= 1 - ((1 - \mu_A(x)) \cdot (1 - \mu_B(x))) \\ &= \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x) \end{aligned}$$

Again, this operator is symmetric, and for  $n$  fuzzy sets  $A_i$ , the degree of membership of an element  $x$  in the union of  $n$  fuzzy sets  $A_i$  is calculated by

$$\mu_{\bigcup_{i=1}^n A_i}(x) := 1 - \prod_{i=1}^n (1 - \mu_{A_i}(x)) \quad (5.2)$$

Figure 5.4 illustrates the algebraic sum by a three-dimensional plot in the interval from zero to one. In comparison to the graph shown in Figure 5.2, there is again no edge of points where  $x = y$ . Furthermore, the truth value  $z = xy$  decreases more slowly with decreasing values of  $x$  and  $y$ .

## 5.2.2 A Compensating Fuzzy Set Connector

The “compensatory and” operator or Gamma-operator has been proposed by (Zimmermann, 1993). It presents a combination of the algebraic product and the algebraic sum. It has been developed to satisfy

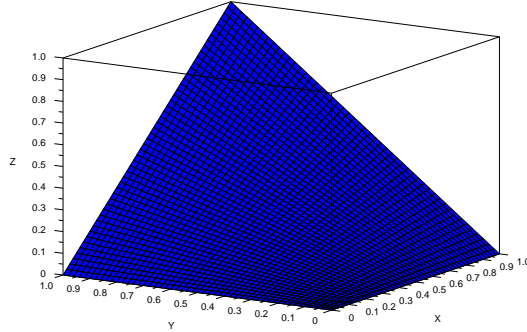


Figure 5.3: Algebraic Product Intersection

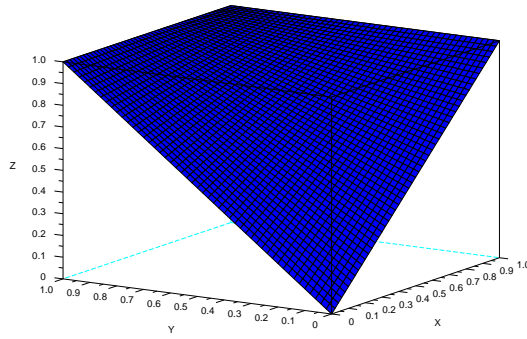


Figure 5.4: Algebraic Sum Union

psychophysical properties of information aggregation in commonsense reasoning. The membership degree of an element  $x$  to the combination of  $n$  fuzzy sets  $A_i$  is defined as follows:

$$\mu_{\Gamma_{i=0}^n A_i}(x) := \left( \prod_{i=1}^n \mu_{A_i}(x) \right)^{(1-\gamma)} \left( 1 - \prod_{i=1}^n (1 - \mu_{A_i}(x)) \right)^{\gamma} \quad (5.3)$$

Depending on the  $\gamma$  parameter, this operator incorporates more of the algebraic product (Equation 5.1) or more of the algebraic sum (Equation 5.2). For  $\gamma = 0$  this operator is equal to the algebraic product, and for  $\gamma = 1$  this operator is equal to the algebraic sum.

The idea behind this operator is an empiric experiment which showed that depending on the context, humans combine the degree of memberships of an element in two fuzzy sets differently, because they vary in compensation. The Gamma operator is able to model this deviant categorization behavior by applying different values for the parameter gamma in different contexts.

Figure 5.5 shows a plot of the connection of two fuzzy membership degrees by the 'compensatory and' operator, with a gamma value of 0.5. As we can see, the form still resembles a conjunction, but the operator compensates lower truth values of  $x$  with greater truth values of  $y$ , and the graph increases

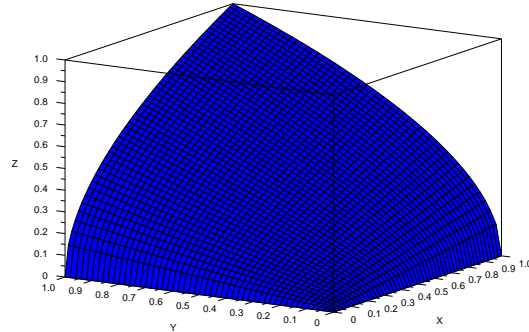


Figure 5.5: *Gamma Operator:  $z = \Gamma(x, y), \gamma = 0.5$*

drastically on the edges.

(Schindler, 1998) used the Gamma-operator for multidimensional tuple classification. The motivation for this was a more intuitively and psychologically accurate combination of tuple classes. In this thesis, the algebraic product is used for set intersection, because of its summarizability, see Chapter 7.5.3.

## 5.3 Fuzzy Logic

This section describes linguistic variables, their description by fuzzy sets and the connection between linguistic terms and fuzzy propositions. Finally, approximate reasoning is described by fuzzy set intersection and union.

### 5.3.1 Linguistic Variables

The concept of linguistic variables provides ergonomic data modeling. It allows to describe measures by linguistic terms. For example, a linguistic variable *age* allows to describe a person's age by the terms *young*, *middle aged*, and *old*. The notion of linguistic variable has been formally defined by Zadeh:

**Definition 3** (Zadeh, 1973) *A linguistic variable is defined by a tuple  $(x, T, U, G, M)$ . In this tuple,  $x$  is the name of the variable.  $T$  is the set of possible linguistic terms for the variable.  $U$  is the definition domain of the underlying base variable  $u$ .  $G$  is a syntactic rule for generating the names of values of  $x$ . Finally,  $M$  is a semantic function associating each value with its meaning.*

For the sake of simplicity, let us define a *linguistic variable* in the following way:

**Definition 4** *A linguistic variable  $V$  on a domain  $D$  is defined by a set of terms*

$$V = \{T_1, \dots, T_n\}$$

Each term  $T_i$  is a fuzzy subset of  $D$ . That is, for every value  $d \in D$ , there is a degree of membership of  $d$  to each term  $T_i$  denoted by

$$\mu_{T_i}(d).$$

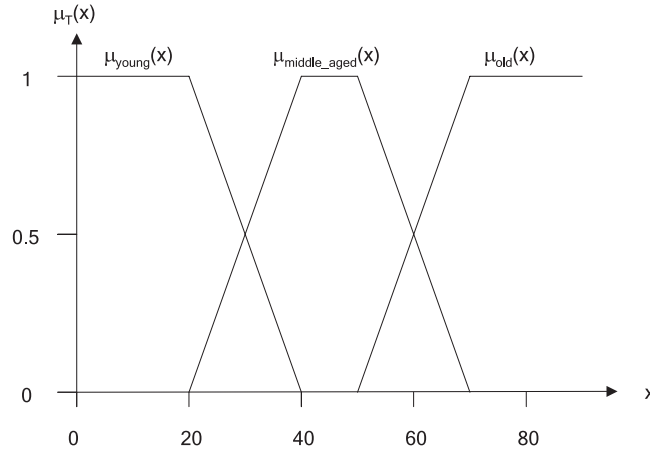


Figure 5.6: *Membership Functions for Terms of a Linguistic Variable*

Consider the previous example of the linguistic variable *age*. The corresponding set of terms is  $age = \{ young, middle\_aged, old \}$ . In Figure 5.6, exemplary membership functions are illustrated. The definition domain is the set of natural numbers,  $D = \mathbb{N}$ . For instance, the degree of membership to the term *old* for an age  $x$  is denoted by  $\mu_{old}(x)$ , and we have  $\mu_{old}(80) = 1$ .

### 5.3.2 Fuzzy Propositions and Approximate Reasoning

In classical propositional logic, truth values are in the set  $\{0, 1\}$ . We can generalize the notion of truth and introduce fuzziness by accepting truth values in the range  $[0, 1]$ . Thus, a proposition  $p$  is neither true nor false, but true *to a certain degree*.

**Definition 5** (Zadeh, 1975) *A fuzzy logic proposition is a statement*

$$p \triangleq x \text{ is } P \quad (5.4)$$

where  $x \in X$  is an element of a universe of discourse  $X$ ,  $P$  is a linguistic term modeled by a fuzzy set, and a membership function

$$\mu_P(x)$$

denotes the degree of membership of  $x$  in the fuzzy subset  $P$ .

For a fuzzy proposition  $p$ , the truth value is defined by

$$T(p) := \mu_P(x)$$

Fuzzy formulae consist of the connection of fuzzy propositions with fuzzy logic operators. Disjunctions and conjunctions of truth values for propositions can be modeled by fuzzy set intersection and union. The semantics of approximate reasoning is expressed by a truth function for formulae. The truth function

$$T : F \mapsto [0, 1]$$

is defined by induction over the truth values for propositions:

- If formula  $a$  is defined by a proposition  $p$ ,  $T(a) := \mu_P(x)$ .
- If  $a$  is a formula,  $T(\neg a) := \mu_{\overline{A}}(x) = 1 - \mu_A(x)$ .
- If  $a$  and  $b$  are formulae,  $T(a \wedge b) = \mu_{A \cap B}(x)$ .
- If  $a$  and  $b$  are formulae,  $T(a \vee b) = \mu_{A \cup B}(x)$ .

Depending on the implementation of fuzzy set union and intersection, the connections of fuzzy propositions yield different values.

## 5.4 Possible Applications of Fuzzy Sets to a Data Warehouse

Two examples of fuzzy set applications to OLAP will be discussed. First, (Laurent, 2002) has introduced the concept of a fuzzy multidimensional database, where every data cell represents a fuzzy set. Second, (Schepperle *et al.*, 2004) describes an idea how the notion of fuzzy contexts could be applied to data aggregation.

### 5.4.1 Fuzzy Multidimensional Data Bases

A framework for fuzzy OLAP has been developed by (Laurent, 2002) and further described by (Laurent, 2003). A fuzzy multidimensional database contains elements  $(v, d)$ , where  $v$  is a value, and  $d \in [0, 1]$  is the confidence degree associated with this value. A domain is a finite set of elements. A dimension is defined on a domain where the pair  $(v(d_i), d(d_i))$  represents element  $d_i$  of dimension  $D_i$ . A fuzzy cube is a mapping

$$D_1 \times \dots \times D_k \mapsto D_C \times [0, 1]$$

where  $D_1, \dots, D_k$  are dimensions, and  $D_C$  is the measure. There is a degree denoting to what extent an element belongs to a cube cell.

In this thesis, a fuzzy OLAP-cube will only contain fuzzy dimension category hierarchies. The underlying elements remain crisp. This will have the advantage of interoperability with relational databases and smaller data volume.

### 5.4.2 Fuzzy Data Aggregation

The concept of fuzzy data aggregation has been described by (Schepperle *et al.*, 2004). The main idea is the categorization of dimensions in an OLAP-cube by fuzzy contexts. This leads to a fuzzy OLAP-cube similar to the approach of (Laurent, 2002). However, the basic granularity of data is crisp, and only the dimension categories are fuzzy.

In an OLAP-cube, a dimension is categorized by one or many categorization levels. Every dimension value category presents an equivalence class of dimension values. As we will see in the following sections, every categorization level can be modeled by a context in the sense of (Schindler, 1998). The idea of (Schepperle *et al.*, 2004) is the fuzzification of these categorization levels by introducing fuzzy contexts on an OLAP-cube's dimensions.

In a crisp dimension, every dimension value belongs to a single dimension category to one hundred percent. In a fuzzy dimension, according to (Schepperle *et al.*, 2004), a dimension value can belong to several fuzzy categories to different degrees. Thus, a fuzzy dimension category is a fuzzy set, and every dimension value has a membership degree in that set. This approach leads to fuzzy dimension hierarchies in an OLAP cube.

In order to use fuzzy contexts for data consolidation, the aggregation functions must be extended. (Schepperle *et al.* , 2004) proposes the following fuzzy set aggregation functions:

- Fuzzy Count: The cardinality of a fuzzy set, the sum of the membership degrees of its elements
- Fuzzy Sum: The sum of elements in a fuzzy set, weighted by their membership degrees
- Fuzzy Average: The fuzzy sum divided by the fuzzy cardinality

## Chapter 6

# Fuzzy Context Classification

The concept of classification as it is understood in this thesis stems from the field of data analysis. Generally, data elements are classified by calculating a membership function for given classes. It is important to note, however, that the notion of *class* in data analysis has nothing to do with the mathematical concept with the same name.

Mathematical class theory is an extension of set theory in order to circumvent certain paradoxes arising from naive set theory. In that theory, all sets are classes, but certain classes are not sets, insofar as they can contain elements, but they cannot be elements themselves. For example, the class of all sets is a proper class. A class in data analysis is rather understood as an equivalence class of data elements. That is why these classes are often called clusters of elements, or element categories.

This section will first introduce the problem of fuzzy classification formally. Then it will explain the underlying mathematical concepts for context classification. Third, the concept of context as described by (Shenoi, 1995) is formally described. Finally, the notion of vague context is introduced according to (Schindler, 1998), and a more rigorous definition of a fuzzy concept is proposed.

### 6.1 Fuzzy Classification

The following section is a summary of the overview given by (Gramann, 1994). Generally, the fuzzy classification problem is formally defined as follows: Let an Object  $O$  be given. This object is characterized by a  $t$ -dimensional feature vector  $\underline{x}_O$  of a universe of discourse  $U$ . A set  $\{c_1, \dots, c_n\}$  of classes is given. The task is to calculate a membership vector  $(m_1, \dots, m_n)$  for the object  $O$ , where  $m_i$  is the degree of membership of  $O$  to class  $c_i$ .

Sometimes, the features of  $\underline{x}_O$  are themselves fuzzy sets. This will be the case in this thesis, when we will calculate the degree of membership of a tuple to a data cell based on fuzzy equivalence classes for attribute values in a fuzzy context. In that case, the fuzzy equivalence class of an attribute will present a fuzzy feature in the sense of (Gramann, 1994).

The classical analytical approach uses a function

$$f : U \mapsto [0, 1]^n$$

mapping a feature vector to a membership vector for the  $n$  classes. In data mining, this function is unknown and it is the task to find this function. In our thesis, this function is known, and the task is simply to classify the objects. These objects will be tuples of a database relation. Furthermore, a data cube will be built based on that classification, where each data cell will present an aggregated *fuzzy tuple*

*class*. Every fact will be calculated by a fuzzy set aggregation of the measure attribute values in a fuzzy equivalence class.

## 6.2 Mathematical Concepts used for Context Classification

This section describes the basic mathematical concepts used for the classification by contexts, which will be used to describe context classification, its fuzzification and the application of fuzzy contexts to OLAP cubes.

- A *partition* of a set  $S$  is a set of subsets  $S_i \subseteq S$  having the following properties:

$$\begin{aligned} 1.) & \quad \bigcup_i S_i = S \\ 2.) & \quad \bigcap_i S_i = \emptyset \end{aligned}$$

Every  $S_i$  will be called a *category* of the partition.

- A relation  $R$  on a set  $S$  is an *equivalence relation* if for all  $a, b$  and  $c$ ,

$$\begin{aligned} 1.) & \quad (a, a) \in R \\ 2.) & \quad (a, b) \in R \Rightarrow (b, a) \in R \\ 3.) & \quad (a, b) \in R \wedge (b, c) \in R \Rightarrow (a, c) \in R \end{aligned}$$

Usually, the fact that  $(a, b) \in R$  is denoted by  $a \sim b$ . The set of all equivalence classes for a set  $S$  and an equivalence relation  $\sim$  is called the *quotient space*  $S / \sim$ , and an equivalence class for an element  $e \in S$  is the set

$$[e] = \{e' \in S \mid e \sim e'\}.$$

(Nguyen & Walker, 1997) proves that for every quotient space, there exists a partition, and for every partition, there exists a quotient space. More specifically, for every partition there is an equivalence relation whose quotient space is equal to that partition. In that case, every category of the partition presents an equivalence class of the corresponding quotient space of the equivalence relation.

- A *fuzzy partition* on a set  $S$  is a set of  $n$  fuzzy subsets  $P = \{S_1, \dots, S_n\}$  satisfying the property

$$\forall x \in S \quad \sum_{S_i \in P} \mu_{S_i}(x) = 1.$$

We will call every member  $S_i$  of that fuzzy partition a *fuzzy category*.

## 6.3 Context Classification

Sujeet Shenoj proposes contexts as a means for information clouding in databases in (Shenoj, 1995). Contexts are defined in terms of equivalence relations:

**Definition 6** (Shenoj, 1995) *A context  $C$  is a partition of a domain  $D$  defined by an equivalence relation  $\equiv_C$ . This relation classifies the elements of  $D$  by a set of complete and mutually disjoint equivalence classes  $c_i$ .*

Since there is a one-to-one mapping between a partition and an equivalence relation, we can define a context  $C$  for a domain  $D$  by a *partition*  $C = \{c_1, \dots, c_n\}$  of that domain, and *only then* we derive a corresponding equivalence relation  $\equiv_C$  defined by

$$x \equiv_C y \Leftrightarrow C(x) = C(y)$$

where  $C(x)$  denotes the partition category  $c_i \in C$  such that  $x \in c_i$ . Two elements are equivalent in a context  $C$  if they are in the same category of the partition  $C$ . Accordingly, we define the equivalence class of an element  $x$  in a context  $C$  by

$$[x]_C = \{x' \in D \mid x \equiv_C x'\}.$$

In relations, every attribute has a domain. Accordingly, the domain of every attribute can be classified by one or more contexts. Thus, in a specific context, every element  $d$  of an attribute's domain  $D$  belongs to a specific category  $c_i = C(d)$ .

**Example 2** Consider the relation described in Table 4.1. Figure 6.1 shows an illustration of context classification of the relation's attributes. Every context is a partition of the attribute's domain. For example, the domain of attribute *bmi* can be defined as

$$\text{dom}(bmi) = [0, 200].$$

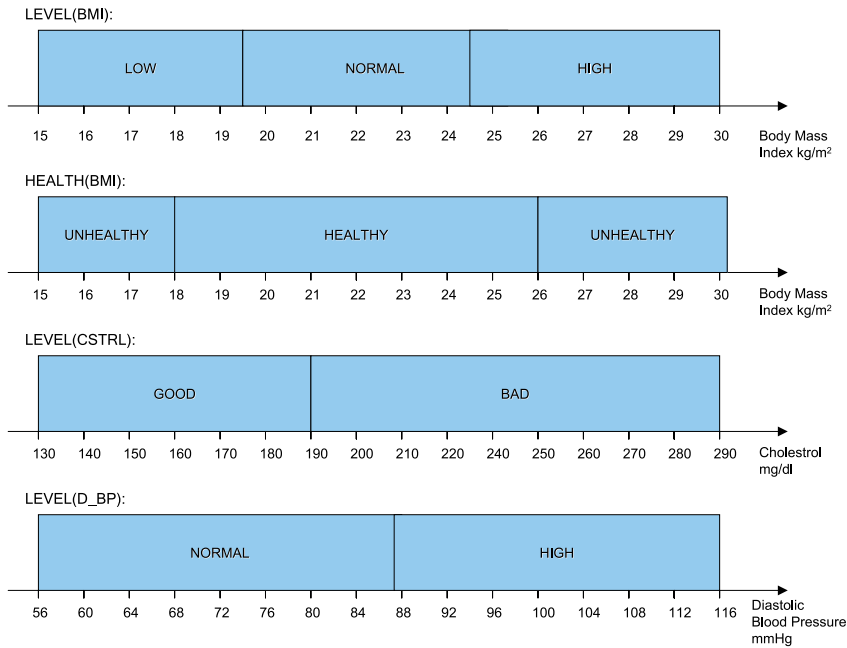


Figure 6.1: Example Contexts Defined on the Domain of the Attributes of Relation 'patient' (Table 4.1)

A possible partition of that domain could be defined in the following way:

$$\text{level}(bmi(t)) = \begin{cases} \text{low} & \text{if } bmi(t) < 19.5 \\ \text{high} & \text{if } bmi(t) > 24.5 \\ \text{normal} & \text{else.} \end{cases}$$

The other contexts  $health(bmi)$ ,  $level(cstrol)$  and  $level(d\_bp)$  shown in Figure 6.1 are defined correspondingly.

Shenoi continues by defining a context's granularity as the coarseness of the equivalence relation. The coarseness depends on the number of equivalences captured by the equivalence relation. Accordingly, contexts can be ordered. The finest context is generated by an equality equivalence relation, where each equivalence class is a singleton. The coarsest context for a domain  $D$  is generated by an equivalence relation  $R = D \times D$ , where there is only one equivalence class which is equal to the domain itself.

(Shenoi, 1995) introduces contexts in order to cloud information for data security. For example, instead of giving an accurate salary value  $Salary = 30'000$ , only an equivalence class is shown. For example, the salary value can be clouded by a fuzzy logic proposition  $Salary \text{ is modest}$ . However, the concept can also be used to consolidate the density of information in data analysis, which will be the case in this thesis.

There is a correspondence between functional dependence and contexts. Every context for an attribute in a relation presents a categorization of that attribute, and the equivalence classes of that context form the domain of a new attribute which is functionally dependent.

*Theorem.* If an attribute  $B$  functionally depends on another attribute  $A$ , then  $dom(B)$  is a context for  $dom(A)$ . Proof: Since  $A \rightarrow B$ , there is a function  $f : dom(A) \mapsto dom(B)$  such that for every element  $b \in dom(B)$  there is an  $a \in dom(A)$  with  $b = f(a)$ . We introduce the equivalence relation  $\equiv_f$  on the set  $dom(A)$  defined by

$$a \equiv_f a' \Leftrightarrow f(a) = f(a').$$

The quotient space of  $\equiv_f$  forms a partition of  $dom(A)$ , where each element in  $dom(B)$  is an equivalence class in the context  $B$ .

As we have seen in Section 4.3.2, dimension category hierarchies can be expressed by functional dependency  $\rightarrow$ . The previous theorem implies that all functional dependencies correspond to a context. Therefore, for modeling dimension category hierarchies, the concepts of functional dependency and context are equivalent.

## 6.4 Fuzzy Context Classification

Whereas (Shenoi, 1995) uses fuzzy sets for modeling contexts, only the  $\alpha_{1,0}$ -cut of trapezoidal fuzzy sets is evaluated for calculating equivalence class memberships. In English, a data item is only assigned to a certain category if the corresponding membership degree is equal to 1.0. Shenoi writes that "Information clouding [...] is accomplished by replacing each [...] piece of exact information with a meaningful fuzzy set whose  $\alpha_{1,0}$ -cut is a superset of the singleton set." That is, a data element either fully belongs to a class, or it does not.

A more radical approach to fuzziness is made by (Schindler, 1998). In this dissertation, the notion of equivalence class membership is genuinely fuzzified. The idea is to assign data elements to different classes to a certain degree.

Schindler realized that in Shenoi's model, data elements can be assigned to several different classes. Therefore, an uncertainty arises. "An assignment of objects to many classes [...] with different degree means that one disposes of information how to handle an object primarily. The uncertainty resulting from a multiple assignment would be reduced." Schindler argues that different membership degrees add information as to which class an element belongs more possibly. Thus he introduces the notion of vague context and the corresponding term fuzzy equivalence class as follows:

**Definition 7** (Schindler, 1998) A vague context  $\hat{C}$  over a domain  $D$  is a linguistic variable where the equivalence classes are fuzzy sets  $\tilde{c}_i$ .

The definition of vague context uses linguistic variables whose terms are fuzzy sets. However, for linguistic terms there is no restriction on the distribution of membership degrees.

If membership degrees are normalized, then classification for OLAP is simplified. In order to normalize the membership degrees to the terms of linguistic variables, we introduce the definition of fuzzy contexts using the notion of fuzzy partition:

**Definition 8** A fuzzy context  $\tilde{C}$  is a fuzzy partition of a domain  $D$ .

This partition  $\tilde{C} = \{\tilde{c}_1, \dots, \tilde{c}_n\}$  consists of a set of *fuzzy equivalence classes*  $\tilde{c}_i$ . Being a fuzzy partition, for every element the sum of membership degrees to all classes adds up to 1. We will present a framework for fuzzy classification based on *fuzzy partitions* in section 7.2.

A fuzzy partition corresponds to a normalized linguistic variable defined on the domain of an attribute  $A$ . The equivalence classes are the terms of the linguistic variable, and for every value  $v \in \text{dom}(A)$  the sum of membership degrees to all terms is 1.

$$\sum_{i=1}^n \mu_{\tilde{c}_i}(v) = 1.$$

**Example 3** The contexts on the attributes of relation *patient* (see Table 4.1) can be fuzzified. For example, as shown in Figure 6.2, *f\_level* is a fuzzy context

$$f\_level = \{low, normal, high\}$$

on attribute *bmi*. The terms are *low*, *normal* and *high* and the corresponding membership functions add up to 1 for every value of *bmi*. Figure 6.3 shows a second fuzzy context *f\_health* on *bmi*, with only two terms, *healthy* and *unhealthy*. Figures 6.4 and 6.5 show the membership functions of fuzzy contexts on the attributes *ctrl* and *d\_bp* respectively.

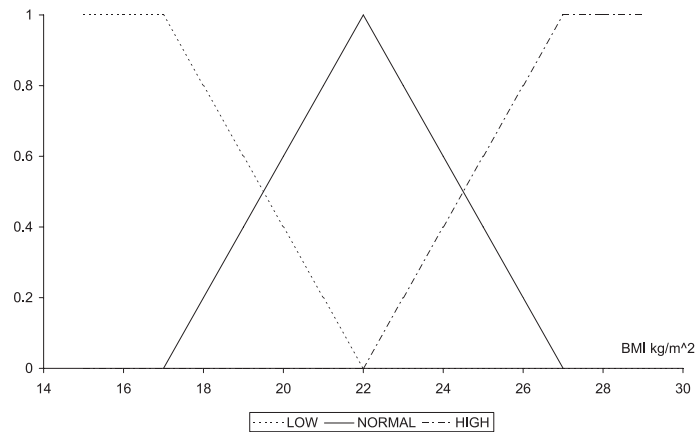


Figure 6.2:  $f_{level}(bmi)$ : A Fuzzy Context on Attribute 'bmi'

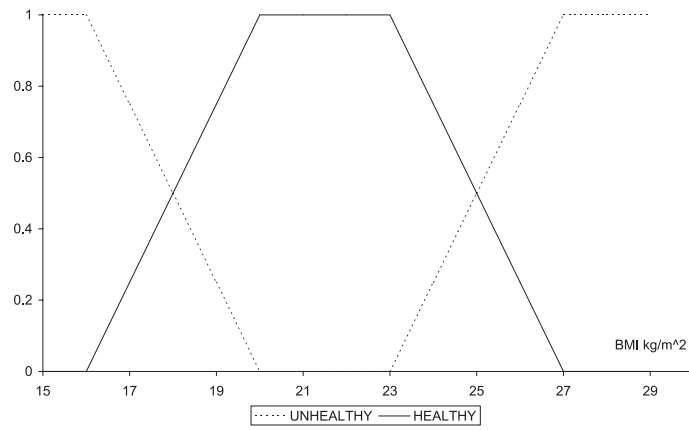


Figure 6.3:  $f_{health}(bmi)$ : A second Fuzzy Context on Attribute 'bmi'

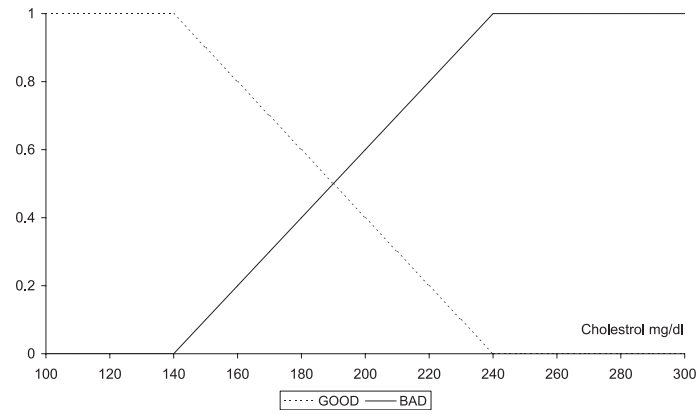


Figure 6.4:  $f_{level}(cstl)$ : A Fuzzy Context on Attribute 'cstl'

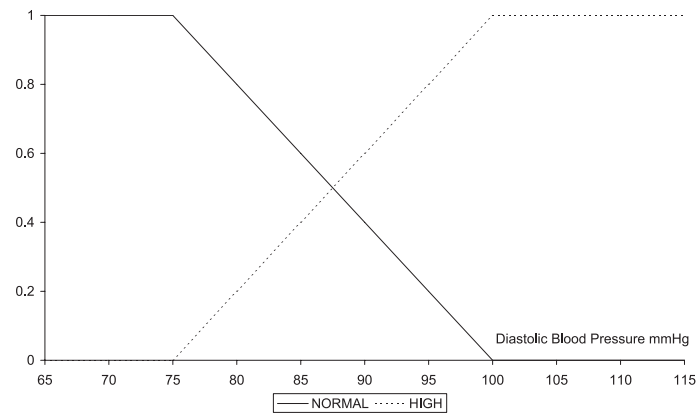


Figure 6.5:  $f_{level}(d\_bp)$ : A Fuzzy Context on Attribute 'd\_bp'

## **Part III**

# **Fuzzy Classification in Online Analytical Processing**

## Chapter 7

# A Framework for Fuzzy Data Consolidation

To consolidate data means to summarize large amounts of data elements in order to extract meaningful statements (propositions). This is usually done by summarizing many data elements in classes or categories. Fuzzy data consolidation is achieved by assigning data elements to fuzzy categories with a certain membership degree. This leads to a more natural assignment of data elements to categories.

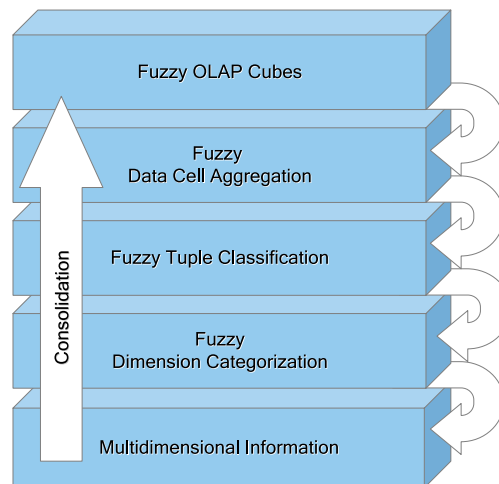


Figure 7.1: A framework for fuzzy data consolidation

The general framework for fuzzy data consolidation consists of different levels of consolidating multidimensional information. Figure 7.1 shows those five levels, where every level depends on the level below itself.

On the basic level, crisp multidimensional information is stored in its finest granularity. On the second level, dimensional attributes are assigned to (possibly fuzzy) context categories. On the third level, multidimensional data elements (tuples) are grouped by classes, where the fuzzy context categories of level 2 serve as classification features. For every combination of dimension values, there is a tuple class. On the fourth level, tuple classes are aggregated using standard relational aggregation functions

extended to fuzzy relations. These aggregations will present the facts in the data cells of a fuzzy OLAP cube. On the top level, the framework will provide OLAP cubes with operators for the manipulation and navigation of fuzzy-summarized data cells.

The process of fuzzy data consolidation allows the creation of fuzzy classes on multidimensional information, as well as the navigation through different levels of aggregation of those classes. The framework will specify the basic concepts of this process.

This chapter first presents the general framework for fuzzy data consolidation. Then, the following sections explain each of the five components of the framework, namely multidimensional information, dimensional categorization, tuple classification, attribute aggregation and OLAP cubes.

## 7.1 Multidimensional Information

In our approach, all *attributes* of a relation in third normal form are dimensions of an information space. The distinction between qualitative dimensions and quantitative measure attributes is only decided during the analysis of data cubes. This approach assumes an underlying multidimensional data domain for the information stored in an OLAP cube. Such a multidimensional domain will be called an information space. A relation can be considered truly multidimensional if its attributes are functionally independent from each other, and if there are no null values in the tuples.

For example, (Skopal *et al.*, 2003) explains multidimensional data elements as points in an  $n$ -dimensional discrete vector space  $\Omega$ , which is a Cartesian product of finite domains  $D_i$ ,  $\Omega = D_1 \times D_2 \times \dots \times D_n$ . A point (tuple) in space is then represented by a vector of coordinates  $d = (d_1, d_2, \dots, d_n)$  where  $d_i \in D_i$ . What (Skopal *et al.*, 2003) call a discrete vector space, is mathematically described by the notion of point lattice.

**Definition 9** (Weisstein, 2005) *A point lattice is a discrete subset of Euclidian  $n$ -dimensional space  $\mathbb{R}^n$ .*

It is a regularly spaced array or grid of points. Figure 7.2 shows an example of a point lattice on a two dimensional Euclidian vector space. Since for all  $n$ ,  $\mathbb{R}^n$  is a vector space, every point in that lattice can be addressed by a linear combination of  $n$  linearly independent vectors from  $\mathbb{R}^n$  and  $n$  natural numbers  $a_i \in \mathbb{N}$ . Thus, using discrete and finite sets of symbols, multidimensional information can be described by a subset of a point lattice.

Consider  $n$  data domains  $D_1, \dots, D_n$  which are sets of elements called symbols. Then a domain  $D'$  is said to be *functionally dependent* of those domains if there is a function mapping

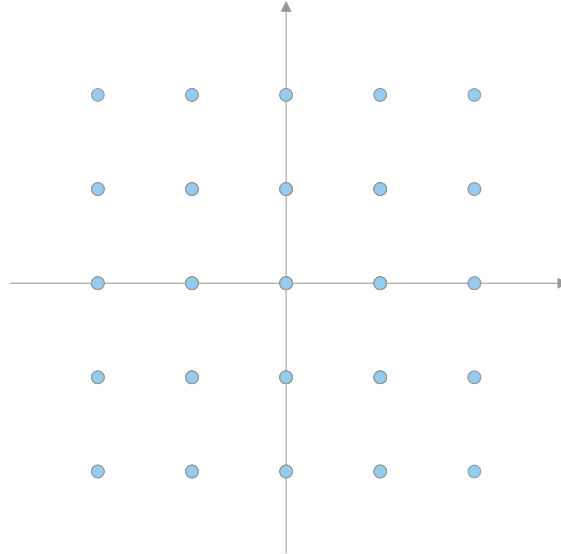
$$f : D_1 \times \dots \times D_n \mapsto D'.$$

A set of  $n$  data domains is called *orthogonal* if no domain is functionally dependent of any other domain. In that case, the relation is in the third normal form, and the identification key encompasses all attributes. We can denote a *Cartesian product of functionally independent data domains as information space*.

**Definition 10** *An  $n$ -dimensional information space  $\Omega$  can then be described by a set  $X$  of  $n$  functionally independent data domains  $X = \{D_1, \dots, D_n\}$  as the Cartesian product*

$$\Omega = D_1 \times \dots \times D_n \tag{7.1}$$

The data domains are called the *dimensions* of the information space. For every information space, any subset of the dimensions will open a subspace. This space encompasses all possible combinations (tuples) of the symbols in the data domains of  $D$ .

Figure 7.2: A Point Lattice on  $\mathbb{R}^2$ 

**Definition 11** A point in an information space is a tuple

$$(d_1, \dots, d_n) \quad d_1 \in D_1, \dots, d_n \in D_n \quad (7.2)$$

In fact, the notion of information space corresponds to a point lattice. Every point  $(i_1, \dots, i_n)$  in the lattice corresponds to a data element  $(d_{i_1}, \dots, d_{i_n})$ . Figure 7.3 shows an illustration of a multidimensional data point as a tuple in a  $n$ -ary relation with orthogonal data domains.

**Definition 12** An information graph is a set of points in an information space.

An information graph corresponds to a set of tuples in the Cartesian product of the data domains. Therefore, it is equivalent to a relation without functional dependencies between its attributes. In terms of predicate calculus, an information graph can be seen as a set of predicate statements in an interpretation - that is, the set of predicate statements that are believed to be true.

**Example 4** *Revisit relation patient (Table 4.1). The attributes describing patient identification, body mass index, cholesterol value, blood pressure and age are functionally independent from one another. Therefore, the domains of patient open a five-dimensional information space. Thus, these attributes of patient are orthogonal and form a multidimensional information space, where the tuples are points in that space, and the attribute values are the coordinates.*

*The relation itself presents a graph as a collection of 25 points in that space. For instance,  $(A, 19.7, 140.7, 92.4, 35)$  is such an information point. In terms of predicate calculus, this point is equivalent to a predicate statement that there exists a patient  $A$  with a body mass index of 19.7, a cholesterol value of 140.7, a diastolic blood pressure of 92.4 and an age of 35.*

To summarize, if  $n$  attribute domains of a relation are functionally independent from each other, then these attributes correspond to a dimension in a point lattice. Furthermore, the tuples of the relation correspond to points in the lattice, and there is a one-to-one mapping between numeric coordinates and indexed data symbols.

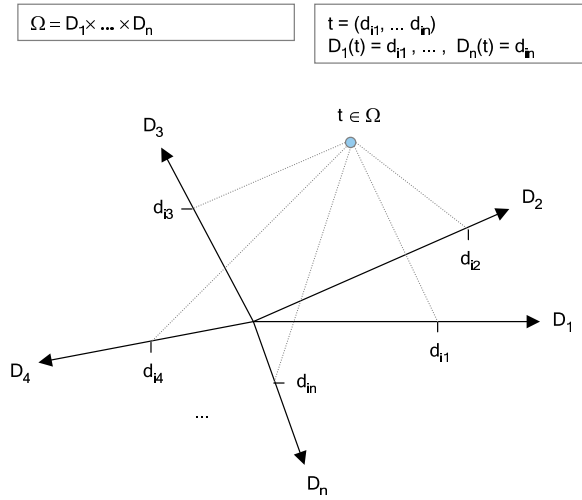


Figure 7.3: Multidimensional Information

## 7.2 Fuzzy Categorization

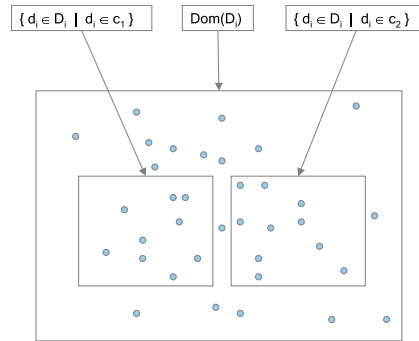


Figure 7.4: Categorization of a Domain

In our model, there are two kinds of classifications: Attribute classification and tuple classification. First, every attribute of a tuple can be classified by one or more contexts, as explained in Section 6.3. Let us call context classification a *categorization* of an attribute. This kind of classification leads to a category hierarchy used for data consolidation in multidimensional analysis. Fuzzy contexts will provide fuzzy categories for fuzzy consolidation paths.

Second, tuples of a relation can be classified according to the values (or categories) of its attributes. The second kind of classification is called *tuple classification* in the sense of section 6.1. In that understanding, every attribute value (or an aggregation of a value) represents a one-dimensional feature of a feature vector consisting of a tuple's attributes. Multidimensional classification uses an  $n$ -dimensional feature vector of memberships to different tuple classes.

As illustrated by Figure 7.4, categorization of dimensional attributes can be accomplished by contexts, where every equivalence class in a context is a category, and every context is a level of categorization. The partial context ordering leads to a category hierarchy for attribute  $A$ . The categorization

of a tuple attribute value  $A(t)$  by a context  $C$  corresponds to the equivalence class  $c_i = [A(t)]_C$  in the context  $C$ . With fuzzy contexts, this classification corresponds to a fuzzy proposition  $x$  is  $\tilde{c}_i$ . For every value  $v$  in a domain  $D$  the function  $\mu_{\tilde{c}_i}(v)$  is the degree of membership of  $v$  to the fuzzy category  $\tilde{c}_i$ .

**Definition 13** A fuzzy categorization of an attribute  $A$  is a fuzzy partition  $\tilde{C} = \{\tilde{c}_1, \dots, \tilde{c}_n\}$  of  $\text{dom}(A)$ , where every  $\tilde{c}_i$  is called a fuzzy category, and the corresponding fuzzy context  $\tilde{C}$  is called a categorization level of  $A$ . For every element  $e$  in the domain of  $A$ , and for every category  $\tilde{c}_i$  in the context  $\tilde{C}$ , the function

$$\mu_{\tilde{c}_i}(e)$$

denotes the degree of membership of  $e$  in that category.

Table 7.1: Example categorizations of the attributes of relation 'patient'

#	patient_id	bmi	cstfr	d_bp	age	health(bmi)	level(bmi)	level(cstfr)	level(d_bp)	f_level(bmi)		f_level(cstfr)			f_level(d_bp)			
										healthy	unhealthy	low	normal	high	good	bad	high	normal
1	A	19.7	140.7	92.4	35	healthy	normal	good	high	0.925	0.075	0.46	0.54	0	0.993	0.007	0.696	0.304
2	B	19.7	172.5	78.2	65	healthy	normal	good	normal	0.925	0.075	0.46	0.54	0	0.675	0.325	0.128	0.872
3	A	19.3	125.4	73.8	36	healthy	low	good	normal	0.825	0.175	0.54	0.46	0	1	0	0	1
4	B	19.9	129.1	74	66	healthy	normal	good	normal	0.975	0.025	0.42	0.58	0	1	0	0	1
5	C	28.7	167.5	115.2	26	unhealthy	high	good	high	0	1	0	0	1	0.725	0.275	1	0
6	D	27.1	283.7	96.5	64	unhealthy	high	bad	high	0	1	0	0	1	0	1	0.86	0.14
7	E	17.2	142.2	91.8	45	unhealthy	low	good	high	0.3	0.7	0.96	0.04	0	0.978	0.022	0.672	0.328
8	F	19.25	160.3	75.4	34	healthy	low	good	normal	0.8125	0.188	0.55	0.45	0	0.797	0.203	0.016	0.984
9	G	15.8	233.6	94.6	76	unhealthy	low	bad	high	0	1	1	0	0	0.064	0.936	0.784	0.216
10	H	22.4	167.2	107.8	72	healthy	normal	good	high	1	0	0	0.92	0.08	0.728	0.272	1	0
11	I	21.6	228.3	84.5	48	healthy	normal	bad	normal	1	0	0.08	0.92	0	0.117	0.883	0.38	0.62
12	J	24.2	256.9	68.6	94	healthy	normal	bad	normal	0.7	0.3	0	0.56	0.44	0	1	0	1
13	K	20.9	144.5	93.7	59	healthy	normal	good	high	1	0	0.22	0.78	0	0.955	0.045	0.748	0.252
14	L	22.5	187.1	85.4	56	healthy	normal	good	normal	1	0	0	0.9	0.1	0.529	0.471	0.416	0.584
15	M	18.4	238.9	87.8	34	healthy	low	bad	high	0.6	0.4	0.72	0.28	0	0.011	0.989	0.512	0.488
16	N	23.2	174.7	86.7	56	healthy	normal	good	normal	0.95	0.05	0	0.76	0.24	0.653	0.347	0.468	0.532
17	O	15.8	172.8	102.3	67	unhealthy	low	good	high	0	1	1	0	0	0.672	0.328	1	0
18	P	27.6	227.4	73.6	34	unhealthy	high	bad	normal	0	1	0	0	1	0.126	0.874	0	1
19	Q	18.5	245.2	71.9	78	healthy	low	bad	normal	0.625	0.375	0.7	0.3	0	0	1	0	1
20	R	18.7	244.7	73.4	45	healthy	low	bad	normal	0.675	0.325	0.66	0.34	0	0	1	0	1
21	S	15.3	278.9	107.6	23	unhealthy	low	bad	high	0	1	1	0	0	0	1	1	0
22	T	28.6	283.4	78.7	87	unhealthy	high	bad	normal	0	1	0	0	1	0	1	0.148	0.852
23	U	29.5	288.7	99.8	58	unhealthy	high	bad	high	0	1	0	0	1	0	1	0.992	0.008
24	V	25.9	189.9	87.4	91	healthy	high	good	normal	0.275	0.725	0	0.22	0.78	0.501	0.499	0.496	0.504
25	W	26	190	87.5	19	unhealthy	high	bad	high	0.25	0.75	0	0.2	0.8	0.5	0.5	0.5	0.5

**Example 5** Consider relation *patient* introduced in Section 4.3.1 and the crisp and fuzzy contexts in Sections 6.4 and 6.3 respectively. Table 7.1 shows that relation together with the categorizations of its attributes, based on the contexts discussed in Examples 2 and 3.

For example, the fuzzy context  $f\_level(bmi)$  presents a fuzzy categorization of attribute *bmi*, where *low*, *normal* and *high* present fuzzy categories. For patient *W*, the body mass index value is  $bmi(A) = 26$ . The fuzzy categorizations of that value using the fuzzy context  $f\_level(bmi)$  are the following:

$$\begin{aligned}\mu_{low}(26) &= 0 \\ \mu_{normal}(26) &= 0.2 \\ \mu_{high}(26) &= 0.8\end{aligned}$$

## 7.3 Fuzzy Multidimensional Tuple Classification

Tuples in a relation can be classified as described in Section 6.1. Let  $R$  be a relation with  $n$  attributes  $A_1, \dots, A_n$ . The features of a tuple  $t \in R$  correspond to the attribute values of  $t$ . If  $t$  is a tuple with  $n$  attributes  $A_1, \dots, A_n$ , the classification feature vector for  $t$  is

$$(A_1(t), \dots, A_n(t)).$$

Classification according to multiple attributes is called *multidimensional*. Single-attribute classification is called one-dimensional. Basically, multidimensional tuple classification is based on the intersection of one-dimensional classes. Accordingly, fuzzy multidimensional tuple classification is equivalent to the fuzzy set intersection of the one-dimensional fuzzy classes.

### 7.3.1 Tuple Classes of Basic Granularity

In order to introduce the basic idea of fuzzy multidimensional tuple classification, we begin with the easily understandable notion of classification using basic uncategorized attribute values as classification features.

**Definition 14** For every value  $v$  in the domain of an attribute  $A$ , there is a one-dimensional class of tuples in  $R$  having the same value for attribute  $A$ .

$$[A \text{ is } v] := \{ t \in R \mid A(t) = v \} \quad (7.3)$$

**Example 6** In relation *patient* (Table 7.1), the equivalence class of tuples with a body mass index value of 19.7 is

$$[bmi \text{ is } 19.7] = \{ t \in patient \mid bmi(t) = 19.7 \} = \{ 1, 2 \}.$$

That is, the tuples with primary key 1 and 2 are in that equivalence class. Accordingly, the only tuple with a cholesterol value of 140.7 is tuple number 1:

$$[cstrol \text{ is } 140.7] = \{ 1 \}.$$

Figure 7.5 shows an example of tuple classification based on two attributes  $A$  and  $B$ . For two values the two-dimensional tuple class  $[A \text{ is } v, B \text{ is } w]$  is equal to the intersection of  $[A \text{ is } v]$  and  $[B \text{ is } w]$ . Consequently, for  $n$  values from the domain of  $n$  attributes  $v_1 \in dom(A_1), \dots, v_n \in dom(A_n)$ , the multidimensional equivalence class of tuples having the same values is equal to the intersection of the corresponding one-dimensional tuple classes.

**Definition 15** A multidimensional tuple class for  $n$  values  $v_1 \in dom(A_1), \dots, v_n \in dom(A_n)$  is defined by

$$[A_1 \text{ is } v_1, \dots, A_n \text{ is } v_n] := \bigcap_{i=1}^n [A_i \text{ is } v_i] = \bigcap_{i=1}^n \{ t \in R \mid A_i(t) = v_i \} \quad (7.4)$$

**Example 7** In our example relation (Table 7.1), the two-dimensional class of tuples having a bmi of 19.7 and a cholesterol level of 140.7 is calculated in the following way:

$$[bmi \text{ is } 19.1, cstrol \text{ is } 140.7] = [bmi \text{ is } 19.7] \cap [cstrol \text{ is } 140.7] = \{ 1 \}.$$

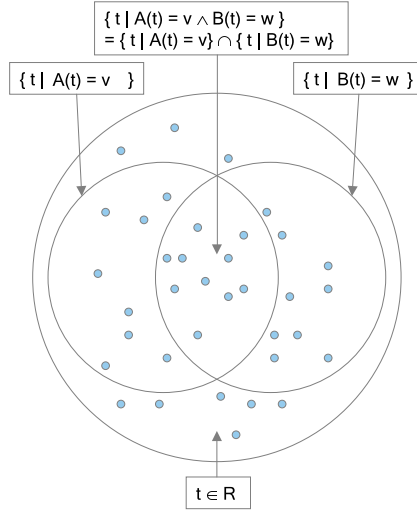


Figure 7.5: Classification of Tuples based on Attribute Values

### 7.3.2 Categorized Tuple Classes

If attributes are categorized by contexts, the classification features are the context categories. The categorized features of a tuple object  $t \in R$  correspond now to categories in a context. If  $t$  is a tuple with  $n$  attributes  $A_1, \dots, A_n$  with corresponding contexts  $C_1, \dots, C_n$  the classification feature vector for  $t$  is

$$(C_1(A_1(t)), \dots, C_n(A_n(t))).$$

**Definition 16** A one-dimensional categorized tuple class defined by a category  $c \in C$  for an attribute  $A$  and a context  $C$  is a set of tuples whose attribute values are in the same category:

$$[A \text{ is } c] := \{ t \in R \mid A(t) \in c \} \quad (7.5)$$

**Example 8** In relation *patient* (Table 7.1), the equivalence class of tuples having a body mass index which is categorized by the context level as *normal* is the following set:

$$\begin{aligned} [bmi \text{ is normal}] &:= \{ t \in patient \mid bmi(t) \in normal \} \\ &= \{1, 2, 4, 10, 11, 12, 13, 14, 16\} \end{aligned}$$

Correspondingly, for attribute *cstrl* and category *good*, we have the following equivalence class:

$$\begin{aligned} [cstrl \text{ is good}] &:= \{ t \in patient \mid cstrl(t) \in good \} \\ &= \{1, 2, 3, 4, 5, 7, 8, 10, 13, 14, 16, 17, 24\} \end{aligned}$$

As illustrated by Figure 7.6, a tuple class for two attribute categories  $c_1, c_2$  is defined as the intersection of the two corresponding one-dimensional tuple classes. In the same way, multidimensional tuple classes can be defined by  $n$  categorized attributes as the intersection of the one-dimensional classes for each context category.

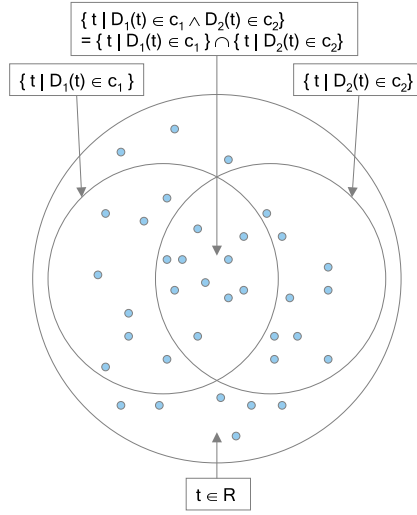


Figure 7.6: Classification of Tuples based on Attribute Categories

**Definition 17** A multidimensional categorized tuple class for  $n$  categories  $c_1, \dots, c_n$  for  $n$  attributes  $A_1, \dots, A_n$  in the contexts  $C_1, \dots, C_n$  is defined by

$$[A_1 \text{ is } c_1, \dots, A_n \text{ is } c_n] := \bigcap_{i=1}^n [A_i \text{ is } c_i] = \bigcap_{i=1}^n \{t \in R \mid A_i(t) \in c_i\} \quad (7.6)$$

**Example 9** In our example relation (Table 7.1), the two-dimensional class of tuples having a bmi in the category normal and a cholesterol level in the category good is calculated in the following way:

$$\begin{aligned} [bmi \text{ is normal}, cstrl \text{ is good}] &= [bmi \text{ is normal}] \cap [cstrl \text{ is good}] \\ &= \{1, 2, 4, 10, 13, 14, 16\} \end{aligned}$$

### 7.3.3 Fuzzy-Categorized Tuple Classes

Features used for classification can be categorized by fuzzy contexts. In that case, the classification features are fuzzified, as described in Section 6.1.

**Definition 18** A one-dimensional fuzzy-categorized tuple class defined by a fuzzy category  $\tilde{c} \in \tilde{C}$  for an attribute  $A$  and a fuzzy context  $\tilde{C}$  is a fuzzy set of tuples. The degree of membership of a tuple  $t$  in that fuzzy tuple class is defined by the degree of membership of the attribute value  $A(t)$  in the fuzzy category  $\tilde{c}$ .

$$\mu_{[A \text{ is } \tilde{c}]} = \mu_{\tilde{c}}(A(t)) \quad (7.7)$$

**Example 10** In relation patient (Table 7.1), the degree of membership of tuple 1 in the fuzzy class of tuples with a normal body mass index is

$$\mu_{[bmi \text{ is normal}]}(1) = \mu_{normal}(bmi(1)) = \mu_{normal}(19.7) = 0.54.$$

Accordingly, the fuzzy tuple class is

$$\begin{aligned} & [bmi \text{ is normal}] \\ = & \{ 1/0.54, 2/0.54, 3/0.46, 4/0.58, 7/0.04, 8/0.45, 10/0.92, 11/0.92, 12/0.56, \\ & 13/0.78, 14/0.9, 15/0.28, 16/0.76, 19/0.3, 20/0.34, 24/0.22, 25/0.2 \} \end{aligned}$$

Correspondingly, for the context  $f\_level(cstrol)$  and fuzzy category good, we have the following fuzzy equivalence class:

$$\begin{aligned} [cstrol \text{ is good}] = \{ & 1/0.993, 2/0.675, 3/1, 5/0.725, 6/1, 7/0.978, 8/0.797, 9/0.064, \\ & 10/0.728, 11/0.117, 12/1, 13/0.955, 4/0.529, 15/0.011, 16/0.653, 17/0.672, \\ & 18/0.126, 19/1, 20/1, 21/1, 22/1, 23/1, 24/0.501, 25/0.5 \} \end{aligned}$$

In analogy to the crisp case, multidimensional fuzzy tuple classes can be defined by  $n$  fuzzy categories as the intersection of the one-dimensional fuzzy classes.

**Definition 19** A multidimensional fuzzy-categorized tuple class for  $n$  fuzzy categories  $\tilde{c}_1, \dots, \tilde{c}_n$  for  $n$  attributes  $A_1, \dots, A_n$  in the fuzzy contexts  $\tilde{C}_1, \dots, \tilde{C}_n$  is a fuzzy set, where each tuple  $t$  has the following membership degree:

$$\begin{aligned} & \mu_{[A_1 \text{ is } \tilde{c}_1, \dots, A_n \text{ is } \tilde{c}_n]}(t) \\ & := \mu_{\bigcap_{i=1}^n [A_i \text{ is } \tilde{c}_i]}(t) \end{aligned} \quad (7.8)$$

Depending the implementation of the fuzzy set intersection, this membership degree is different. (Schindler, 1998) uses the Gamma-operator (see Section 5.2.2) for this intersection. However, formally the Gamma-operator is not an intersection operator, but a mixture of intersection and union. Furthermore, the resulting set intersections are not summarizable, that is, *the union of the intersections does not return the original fuzzy set*. Consequently, the sum of membership degrees of tuples to the different tuple classes does not add up to one.

Contrarily, the algebraic product set intersection yields fully summarizable sets, where the sum of membership degrees to the individual tuple classes always add up to one. Using the algebraic product for set intersection, the membership degree of a tuple in a fuzzy class is therefore

$$\begin{aligned} & \mu_{\bigcap_{i=1}^n [A_i \text{ is } \tilde{c}_i]}(t) \\ & = \prod_{i=1}^n \mu_{[A_i \text{ is } \tilde{c}_i]}(t) \\ & = \prod_{i=1}^n \mu_{\tilde{c}_i}(A_i(t)) \end{aligned} \quad (7.9)$$

**Example 11** In our example relation (Table 7.1), the degree of membership of tuple 2 in the fuzzy class of tuples with a normal body mass index and a good cholesterol value is calculated in the following way:

$$\begin{aligned} & \mu_{[bmi \text{ is normal}, cstrol \text{ is good}]}(1) \\ & \mu_{[bmi \text{ is normal}] \cap [cstrol \text{ is good}]}(1) \\ & = \mu_{[bmi \text{ is normal}]}(1) \cdot \mu_{[cstrol \text{ is good}]}(1) \\ & = 0.54 \cdot 0.993 = 0.53622 \end{aligned}$$

## 7.4 Fuzzy Data Aggregation

Aggregation of crisp information is a usual process in relational data bases. Relations, that is, sets of tuples, are aggregated with respect to a particular attribute and aggregation function. The usual

aggregation operators are sum, average, count, minimum and maximum. Accordingly, a scalar value is calculated based on the attribute values in a relation. For example, a set of tuples  $R = \{t_1, \dots, t_n\}$  can be aggregated using the aggregation function  $sum$  on an attribute  $A_1$ . In that case, the aggregation value is calculated by

$$f_{A_1}(R) = sum_{A_1}(R) = \sum_{t \in R} A_1(t).$$

**Definition 20** An aggregation in an  $n$ -dimensional domain  $\Omega = D_1 \times \dots \times D_n$  is a mapping

$$f_A : P(\Omega) \mapsto V,$$

where  $f$  is an aggregation function,  $A$  is an attribute,  $P(\Omega)$  is the set of all possible relations in  $\Omega$ , and  $V$  is a set of aggregate values.

A tuple class can be aggregated by an aggregation function as well. This aggregation of fuzzy tuple classes will be used to calculate facts in fuzzy OLAP cubes.

**Example 12** In relation patient (Table 7.1), the average age in the class

$$[bmi \text{ is normal, } cstrl \text{ is good}] = \{1, 2, 10, 13, 14, 16\}$$

is an aggregation

$$avg_{AGE}([bmi \text{ is normal, } cstrl \text{ is good}]) = 57, \bar{16}.$$

The aggregation of attribute values in a fuzzy relation is called *fuzzy aggregation*, as described by (Rundensteiner & Bic, 1992) and (Schepperle *et al.*, 2004). Our approach will use the propositions of (Schepperle *et al.*, 2004), because they provide a defuzzification of fact values, which is more ergonomic for the user. Fuzzy linguistic variables are often easily understood by users, but membership degrees are less intuitive. Thus, aggregation of fuzzy information presents a *defuzzification* of fuzzy data in order to be understandable for the user. Basically, a fuzzy aggregation is the calculation of a scalar value based on a fuzzy set of tuples (a fuzzy relation). More formally:

**Definition 21** A fuzzy aggregation in an  $n$ -dimensional domain  $\Omega = D_1 \times \dots \times D_n$  is a mapping

$$f_A : F(\Omega) \mapsto V,$$

where  $f$  is a fuzzy aggregation function,  $A$  is an attribute,  $F(\Omega)$  is the set of all possible fuzzy relations in  $\Omega$ , and  $V$  is a set of aggregate values.

The following points define the fuzzy aggregation functions used in this framework:

- *Fuzzy Count:*

As proposed by (Schindler, 1998), this operator calculates the cardinality of the fuzzy set of tuples  $R$  defined by

$$fcount_A(R) = \sum_{t \in \Omega} \mu_R(t) \quad (7.10)$$

- *Fuzzy Sum:*

As proposed by (Schindler, 1998), this operator calculates the fuzzy sum of numeric attribute values in a fuzzy tuple class  $R$  as the sum of values  $A(t)$  weighted by the membership degree of tuple  $t$  in the fuzzy class  $R$ :

$$fsum_A(R) = \sum_{t \in \Omega}^n A(t) \cdot \mu_R(t) \quad (7.11)$$

- *Fuzzy Average:*

As proposed by (Schindler, 1998), this operator calculates the average of numeric attribute values in a fuzzy class  $R$ . This operation corresponds to a combination of fuzzy count and fuzzy sum:

$$favg_A(R) = \frac{fsum_A(R)}{fcount_A(R)} \quad (7.12)$$

- *Fuzzy Minimum:*

This fuzzy minimum is an experimental operator. A value  $v$  is considered the fuzzy minimum if its multiplication with the membership degree of the value in the fuzzy class  $R$  is minimal:

$$fmin_A(R) = arg \min_v \bigcup_{v \in dom(A)} v \cdot \mu_R(v) \quad (7.13)$$

- *Fuzzy Maximum:*

This fuzzy maximum is an experimental operator. A value  $v$  is considered the fuzzy maximum if its multiplication with the membership degree of the value in the fuzzy class  $R$  is maximal:

$$fmax_A(R) = arg \max_v \bigcup_{v \in dom(A)} v \cdot \mu_R(v) \quad (7.14)$$

- *Fuzzy Classification of attribute values:*

This kind of aggregation is a function mapping from a fuzzy relation of tuples  $R$  to a fuzzy set of attribute values  $C$ . This operator returns the fuzzy class of attribute values based on the fuzzy set of tuples  $R$ . For every value in the domain of attribute  $A$ , the  $fclass$  operator calculates a membership degree to a class  $C$  of attribute values. This membership is calculated as the normative sum of all tuple membership values of tuples in the fuzzy class of tuples  $R$ . For every value  $v \in dom(A)$ , the membership degree of  $v$  in class  $C$  is defined as the average membership of the underlying tuples in  $R$ :

$$\begin{aligned} fclass_A : F(\Omega) &\rightarrow F(dom(A)) \\ fclass_A(R) &= C \\ \Leftrightarrow \\ \mu_C(v) &= \frac{1}{N} \sum_{t \in R} \mu_R(t) \end{aligned} \quad (7.15)$$

The  $fclass$  operator corresponds to a fuzzy classification in the sense of (Schindler, 1998), where they use gamma conjunction for the aggregation of membership values. However, we use the algebraic product in order to make classification summarizable. Thus, the aggregated membership corresponds to the average of the single membership values  $\mu_R(t)$ .

## 7.5 Fuzzy OLAP Cubes

An OLAP-cube is a data model which provides a dimensional access to data consolidation. It consists of a grid of *data cells* laid out by the Cartesian product of the dimensions. There is one data cell for every combination of dimension values. Every cell classifies the tuples in a base relation by their dimension values, or their (possibly fuzzy) dimension categories. Every cell contains a *fact*, which presents an aggregation of the multidimensional tuple class defined by the dimension values. A *fact* is an aggregation of the measure attribute's values for each tuple in a data cell using a set aggregation function.

### 7.5.1 OLAP Cubes with Crisp Dimension Categorization

An OLAP cube on a relation is described by a scheme defining the dimensions and measures of a cube. The dimension values of a cube can be categorized by contexts for the dimensional attributes.

Consider an information space

$$\Omega = D_1 \times \dots \times D_n$$

of orthogonal data domains (see Section 7.1). Let  $R \subset \Omega$  be a relation with attributes  $A = \{A_1, \dots, A_n\}$  such that  $\text{dom}(A_i) = D_i$ .

**Definition 22** A categorized OLAP-cube is defined by a scheme

$$C = (R, D, L, M, f) \quad (7.16)$$

where  $D = \{D_1, \dots, D_n\} \subset A$  is a set of dimension attributes from  $R$ ,  $M$  is a measure attribute from  $R$ ,  $f$  is a set aggregation function, and  $L = \{C_1, \dots, C_n\}$  is a set of categorization levels, where every  $C_i$  is the categorization level for dimension  $D_i$ , defined as a context on the underlying attribute.

A measure attribute  $M$  and an aggregation function  $f$  for data aggregation are given. Every dimension  $D_i \in D$  is associated with a context defining the level of categorization. The possible dimension values of a dimension  $D_i$  are categories  $c_i \in C_i$ .

**Example 13** An example of a cube  $C = (R, D, L, M, f)$  can be defined on our example relation  $R = \text{patient}$  (Table 7.1). Its dimensions are the attributes  $D = \{\text{bmi}, \text{cstnl}, \text{d\_bp}\}$ , with associated categorization levels  $L = \{\text{health}(\text{bmi}), \text{level}(\text{cstnl}), \text{level}(\text{d\_bp})\}$ . Its measure attribute is  $M = \text{patient\_id}$ . An aggregation function can be chosen as  $\text{count}$  or  $\text{classify}$ , since the measure is not numeric.

An extension of a categorized cube scheme is a set of facts, one for every data cell. A categorized data cell corresponds to a multidimensional tuple class. The categorizations of dimension values in their associated context level are the classification features. In a categorized data cube, every combination of dimension values  $(c_1, \dots, c_n)$  represents a data cell, to which the following tuple class is associated:

$$[D_1 \text{ is } c_1, \dots, D_n \text{ is } c_n]$$

**Example 14** In our example cube, for example, the combination of dimension values (healthy, good, normal) represents a data cell, which is associated with the tuple class

$$[\text{bmi is healthy}, \text{cstnl is good}, \text{level}(\text{d\_bp}) \text{ is normal}].$$

**Definition 23** A fact in a categorized data cell  $(c_1, \dots, c_n)$  is an aggregation of the tuples in the corresponding tuple class

$$f_M[D_1 \text{ is } c_1, \dots, D_n \text{ is } c_n] \quad (7.17)$$

using an aggregation function  $f$  on the measure attribute  $M$ .

**Example 15** A fact in the data cell (healthy, good, normal) in our example cube, using the  $\text{count}$  aggregation function, shows the number of patients with a healthy body mass index, a good cholesterol value and a normal blood pressure. This fact is calculated by an aggregation

$$\text{count}_{\text{patient\_id}}[\text{bmi is healthy}, \text{cstnl is good}, \text{d\_bp is normal}]$$

Table 7.2: Example cube on relation 'patient' with crisp dimension categorization

	Body Mass Index:	unhealthy	healthy
cholesterol Value:	Blood Pressure:		
good	normal		A B F L N V
good	high	C E O	A H K
bad	normal	P T	I J Q R
bad	high	D G S U W	M

**Example 16** Table 7.2 shows an example of our cube using measure value classification as aggregation function. In this case, the patient id's are classified in the corresponding data cells.

For example, the patients which fit into the class

$$[ \text{bmi is healthy, cstrl is good, d_bp is normal} ]$$

are the patients A, B, F, L, N and V. Patient A is classified in two classes, because he has two tuple entries with different classifications.

Note that patients V and W have almost the same attribute values, but because of crisp classification, patient v is classified in the most healthy class, but patient W is classified in the most endangered class. Thus, very similar attributes at the critical class limits can lead to distortions in crisp classifications. Preventing that distortion is the advantage of fuzzy classification, as we will see in the next example.

A cube of basic granularity has dimension values in domain of the attributes in a relation  $R$ . A data cell of basic granularity is a special case of a categorized class, where the context contains singleton sets as categories, and for all dimension values, the context of basic granularity is chosen as dimension categorization level. The equivalence classes of the basic granularity context are singleton sets of dimension values.

## 7.5.2 OLAP Cubes with Fuzzy Dimension Categorization

This subsection presents the main idea of the thesis. Dimension categories in an OLAP cube can be fuzzified using fuzzy contexts. For every dimension, a fuzzy categorization level is associated. This leads to fuzzy data cells, which are aggregated to facts using aggregation functions on fuzzy relations.

**Definition 24** A fuzzy categorized OLAP-cube is defined by a scheme

$$C = (R, D, \tilde{L}, M, f) \quad (7.18)$$

where  $R$  is a relation  $R$  with a set of attributes  $A$ ,  $D = \{D_1, \dots, D_n\} \subset A$  is a set of dimension attributes of  $R$ ,  $M$  is the measure attribute of  $R$ ,  $f$  is a fuzzy set aggregation function, and  $\tilde{L} = \{\tilde{C}_1, \dots, \tilde{C}_n\}$  denotes the consolidation level of the cube, where every  $\tilde{C}_i \in \tilde{C}(D_i)$  is a fuzzy context presenting the associated consolidation level for dimension  $D_i$ .

**Example 17** An example of a cube  $C = (R, D, \tilde{L}, M, f)$  can be defined on our example relation  $R = \text{patient}$  (Table 7.1). Its dimensions are the attributes  $D = \{\text{bmi}, \text{cstrl}, \text{d_bp}\}$ , with associated fuzzy categorization levels  $\tilde{L} = \{f\_health(\text{bmi}), f\_level(\text{cstrl}), f\_level(\text{d_bp})\}$ . Its measure attribute is  $M = \text{patient\_id}$ . Only the functions  $fcount$  or  $fclassify$  can be chosen as aggregation function, since the measure is not numeric.

An extension of a categorized cube scheme is a set of data cells containing facts. A fuzzy categorized data cell is a multidimensional fuzzy tuple class. The fuzzy categorization of a dimension's domain in their associated context level are the classification features. In a fuzzy OLAP cube, every combination of fuzzy categorized dimension values  $(\tilde{c}_1, \dots, \tilde{c}_n)$  represents a data cell, to which a tuple class  $[D_1 \text{ is } \tilde{c}_1, \dots, D_n \text{ is } \tilde{c}_n]$  in relation  $R$  is associated.

**Example 18** *In our example cube, the combination of dimension values (healthy, good, normal) represents a data cell, which is associated with the fuzzy tuple class*

$$[ \text{bmi is healthy, cstrl is good, d_bp is normal} ].$$

In an extension of such a scheme, every dimension  $D_i \in D$  is associated with a fuzzy context  $\tilde{C}_i$  called the dimension's fuzzy categorization level. For every value  $v \in \text{dom}(A_i)$ , and every fuzzy category  $\tilde{c}_i$ , the function

$$\mu_{\tilde{c}_i}(v) \tag{7.19}$$

denotes the degree of membership of the attribute value in the fuzzy category.

**Definition 25** *A fact in a fuzzy categorized data cell is a fuzzy set aggregation of the tuples in the corresponding tuple class*

$$f_M[D_1 \text{ is } \tilde{c}_1, \dots, D_n \text{ is } \tilde{c}_n] \tag{7.20}$$

using an aggregation function  $f$  on the measure attribute  $M$ .

**Example 19** *The fact in data cell (healthy, good, normal) in our example cube, using the fcount aggregation function, shows the number of patients with a healthy body mass index, a good cholesterol value and a normal blood pressure. This fact is calculated by an aggregation*

$$f_{\text{count}_{\text{patient\_id}}}[ \text{bmi is healthy, cstrl is good, d_bp is normal} ]$$

For tuples  $t \in R$ , the membership degree in a data cell  $(\tilde{c}_1, \dots, \tilde{c}_n)$  is defined by the membership degree of  $t$  in the corresponding tuple class.

$$\mu_{(\tilde{c}_1, \dots, \tilde{c}_n)}(t) = \mu_{\bigcap_{i=1}^n [D_i \text{ is } \tilde{c}_i]}(t) \tag{7.21}$$

In this model, we take the algebraic product for the intersection of fuzzy sets. Thus, the membership degree of a tuple  $t$  to a fuzzy data cell  $(\tilde{c}_1, \dots, \tilde{c}_n)$  is calculated by a multiplication of membership degrees of the dimension attribute values  $D_i(t)$  to the category  $\tilde{c}_i$ .

$$\mu_{(\tilde{c}_1, \dots, \tilde{c}_n)}(t) = \prod_{i=1}^n \mu_{\tilde{c}_i}(D_i(t)) \tag{7.22}$$

A crisp context presents a special case of fuzzy context. In this case, the membership degree of tuple  $t$  to a tuple class  $[D_i \text{ is } c_i]$  will be 1 if and only if the value is in the category :

$$\mu_{c_i}(A_i(t)) := \begin{cases} 1 & \text{if } D_i(t) \in c_i \\ 0 & \text{else.} \end{cases}$$

**Example 20** *Table 7.3 shows the example cube using fuzzy measure value classification as fact aggregation. For example, patient A belongs to data cell (healthy, good, normal) to a degree of 0.552. This degree is calculated in the following way: There are two tuples for patient A in Table 7.1, namely the tuples 1 and 3. Using the algebraic product, the membership degree of tuple 1 is*

$$\mu_{(\text{healthy,good,normal})}(1) = \mu_{\text{healthy}}(19.7) \cdot \mu_{\text{good}}(140.7) \cdot \mu_{\text{normal}}(92.4) = 0.925 \cdot 0.993 \cdot 0.304 \approx 0.279.$$

The membership degree of tuple 3 is calculated analogically, and yields 0.825. The classification of patient A is calculated using the average of the two membership degrees:

$$\mu_{(\text{healthy,good,normal})}(A) = (0.279 + 0.825)/2 = 0.552$$

Table 7.3: Example cube on relation 'patient' with fuzzy dimension categorization

	Body Mass Index:	healthy	unhealthy
cholesterol Value:	Blood Pressure:		
good	normal	(A, 0.552) (B, 0.76) (E, 0.096) (F, 0.637) (I, 0.073) (K, 0.241) (L, 0.309) (M, 0.0030) (N, 0.33) (V, 0.069) (W, 0.063)	(A, 0.099) (B, 0.035) (E, 0.225) (F, 0.147) (G, 0.014) (M, 0.0020) (N, 0.017) (P, 0.126) (V, 0.183) (W, 0.188)
good	high	(A, 0.32) (B, 0.04) (E, 0.197) (F, 0.01) (H, 0.728) (I, 0.044) (K, 0.714) (L, 0.22) (M, 0.0030) (N, 0.29) (V, 0.068) (W, 0.063)	(A, 0.026) (B, 0.0030) (C, 0.725) (E, 0.46) (F, 0.0020) (G, 0.05) (M, 0.0020) (N, 0.015) (O, 0.672) (V, 0.18) (W, 0.188)
bad	normal	(A, 0.0010) (B, 0.131) (E, 0.0020) (F, 0.162) (I, 0.547) (J, 0.7) (K, 0.011) (L, 0.275) (M, 0.29) (N, 0.175) (Q, 0.625) (R, 0.675) (V, 0.069) (W, 0.063)	(A, 0.0) (B, 0.011) (D, 0.14) (E, 0.0050) (F, 0.037) (G, 0.202) (J, 0.3) (M, 0.193) (N, 0.0090) (P, 0.874) (Q, 0.375) (R, 0.325) (T, 0.852) (U, 0.0080) (V, 0.182) (W, 0.188)
bad	high	(A, 0.0020) (B, 0.019) (E, 0.0040) (F, 0.0030) (H, 0.272) (I, 0.336) (K, 0.034) (L, 0.196) (M, 0.304) (N, 0.154) (V, 0.068) (W, 0.063)	(A, 0.0) (B, 0.0020) (C, 0.275) (D, 0.86) (E, 0.01) (F, 0.0010) (G, 0.734) (M, 0.203) (N, 0.0080) (O, 0.328) (S, 1.0) (T, 0.148) (U, 0.992) (V, 0.179) (W, 0.188)

Unlike in the cube with crisp dimension categorization, the patients V and W are not anymore classified in the best and the worst classes respectively. Using fuzzy contexts, those patients are now

classified by several classes to a certain degree. Thus, the fact that patients  $V$  and  $W$  have very similar attributes, and that those attributes are in a critical limit area, does not lead to distortions in the classification anymore.

### 7.5.3 Summarizability

Aggregations in a crisp cube are summarizable if they are complete, disjunctive and type-compatible. (Schepperle *et al.*, 2004) extends this definition to fuzzy contexts, where fuzzy aggregations are fuzzy-summarizable if the sum of all membership degrees of a data elements in all data cells is exactly one.

An interesting property emerged from the application of algebraic product to fuzzy set intersection for the calculation of multidimensional fuzzy tuple classes. If there are  $N$  tuples in  $R$ , and  $K$  fuzzy tuple classes  $R_i$  defined by fuzzy contexts, then

$$\sum_{i=1}^K fcount(R_i) = N$$

This stems from the fact that if there are  $K$  tuple classes  $R_i$  then for a tuple  $t$ ,

$$\sum_{i=1}^K \mu_{R_i}(t) = 1$$

because the tuple classes form a fuzzy partition, whose membership degrees sum up to 1 by definition.

However, using the minimum operator or the Gamma operator, this sum is not equal to 1. This property of the algebraic product intersection has been discovered during the implementation of the prototype. This property can be verified by calculation, but a formal proof is missing.

Note that in the previous example, for every patient, the membership degrees to all classes *always add up to one*. The reason for this normalization is (1) normalized fuzzy context class memberships, and (2) the use of algebraic product for the combination of tuple classification features.

It is easy to show that the gamma operator leads to membership degrees which do not add up to one. This leads to distortions, because the sum of the facts in the data cells do not yield the same result as the sum of the base data. Thus, aggregations based on the Gamma-operator are not summarizable.

**Example 21** For example, using the gamma operator to combine classification features with  $\gamma = 0.5$ , the membership degree of tuple 1 in data cell (healthy, good, normal) is calculated in the following way:

$$\begin{aligned} & \mu_{(healthy,good,normal)}^{\gamma}(1) \\ &= (0.925 \cdot 0.993 \cdot 0.304)^{0.5} \cdot (1 - (1 - 0.925) \cdot (1 - 0.993) \cdot (1 - 0.304))^{0.5} \\ &\approx 0.5283. \end{aligned}$$

The membership degree of tuple 1 in the other data cells are calculated analogically:

$$\begin{aligned} \mu_{(healthy,good,high)}^{\gamma}(1) &= 0.799494446 \\ \mu_{(healthy,bad,normal)}^{\gamma}(1) &= 0.04320149 \\ \mu_{(healthy,bad,high)}^{\gamma}(1) &= 0.066366925 \\ \mu_{(unhealthy,good,normal)}^{\gamma}(1) &= 0.150127841 \\ \mu_{(unhealthy,good,high)}^{\gamma}(1) &= 0.227447947 \\ \mu_{(unhealthy,bad,normal)}^{\gamma}(1) &= 0.007587409 \\ \mu_{(unhealthy,bad,high)}^{\gamma}(1) &= 0.016228641 \end{aligned}$$

Using the gamma operator, the sum of all membership degrees of tuple 1 in all classes is 1.838781841. This leads to distortions. For example, using the count aggregation, this tuple counts as approximately 1.83 tuples. Thus, the facts in the cube are not summarizable, because the sum of facts in the data cell is different from the sum of the base data in the relation. Therefore, in order to calculate facts that are summarizable, we use the algebraic product.

It is possible to make multidimensional fuzzy classification based on the Gamma operator summarizable. In that case, the membership degrees must be normalized. However, it presents a considerable calculation effort, since this normalization must occur for every tuple and for every class.

### 7.5.4 Operators on Fuzzy OLAP Cubes

In order to manipulate OLAP cubes, some basic operators are proposed: the change of aggregation function, slicing, dicing and drilling.

- *Apply:*

This operator applies a different aggregation function  $f'$  to a data cube  $C = (R, D, L, M, f)$ . The resulting cube  $C'$  has  $f'$  as new aggregation function for facts.

$$C' = \text{apply}_{f'}(C) = (R, D, L, M, f')$$

- *Slice:*

Slicing a dimension  $D_i$  of a fuzzy data cube with a dimension value  $c_i$  generates a new cube  $C'$  whose data cells are addressed by that dimension value:

$$C' = \text{slice}_{D_i, c_i}(C) = (R, D', C, M, f)$$

where in  $D'$ , the domain of dimension  $D_i$  is restricted to  $c_i$ .

- *Roll-up / drill down:*

Rolling up a dimension  $D_i$  to a classification level of coarser granularity or drilling down to a classification level of finer granularity generates a new cube  $C'$ ,

$$C' = \text{drill}_{D_i, C'_i}(C) = (R, D, L', M, f)$$

where in  $L'$  the new consolidation level for dimension  $D_i$  is the fuzzy context  $C'_i$ .

- *Dice:*

Dicing a dimension  $D_i$  by a domain restriction  $P$  generates a new cube  $C'$ ,

$$C' = \text{dice}_{D_i, P}(C) = (R, D', L, M, f)$$

where in the new set of dimensions  $D'$  the domain of dimension  $D_i$  is restricted to the values which satisfy  $P$ .

The extension of cubes generated by the application of those Operators has to be recalculated using the framework in Section . Again, fuzzy tuple classes are generated for each data cells, and those classes are aggregated with the aggregation function. But since the cube specification has been altered with an operator, the resulting cube is different.

## Chapter 8

# A Prototype of an OLAP-Tool Supporting Fuzzy Consolidation

In this thesis, a software prototype has been developed which can perform OLAP-operations on data using fuzzy contexts. The approach which inspired the idea behind this is described by (Schepperle *et al.*, 2004). The main concept is to categorize elements in a dimension based on a fuzzy classification of its domains.

The interesting part of the approach is that it combines OLAP data consolidation with the concept of fuzzy classification. Consolidation of base data through fuzzy contexts is described in Section 6.4. In fact, a category hierarchy of a dimension corresponds to a series of classifications of its domain. The application of fuzzy contexts leads to a fuzzy data consolidation path.

The main idea is that the cells of a data cube present multidimensional classifications of the raw data. In FC-OLAP, these data cells have been extended to fuzzy classes. The software is capable of calculating the membership degrees of tuples in a relation to those data cells, and aggregates these data cells with an aggregation function. In fact, a cube in FC-OLAP is a grid where fuzzy classes are laid out according to the dimension values of a cube. Furthermore, FC-OLAP allows the navigation and manipulation of that layout.

The purpose of the prototype implementation is not a fully functional OLAP tool, but a feasibility study. The aim is to show that fuzzy classification can be applied to OLAP cubes not only in theory, but in practice. Consequently, the framework described in Chapter 7 has been implemented in a real software application called FC-OLAP (fuzzy consolidation OLAP) running on a relational database management system. This chapter describes the implementation of the prototype, explains its usage and gives an evaluation and an outlook stating possible improvements.

### 8.1 Implementation

This section describes the implementation of the software prototype FC-OLAP. The purpose of this section is to show system design as well as system architecture of FC-OLAP.

#### 8.1.1 Functionality Specification

The main aim of the FC-OLAP implementation is to provide a look and feel of how fuzzy data classification can be applied to multidimensional data analysis. Figure 8.1 gives an overview of the functionality

of FC-OLAP, as it was specified before the actual programming.

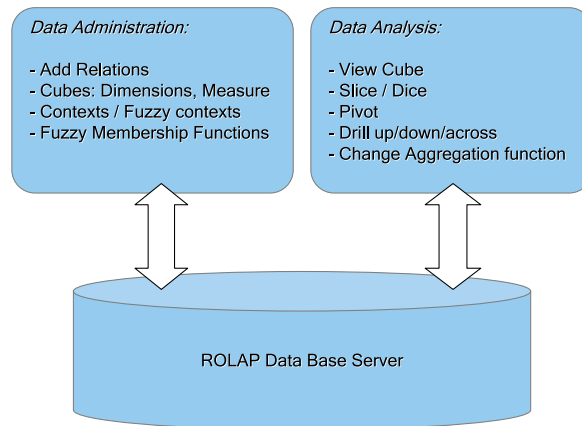


Figure 8.1: FC-OLAP: Functionality Overview

The functionality of an application which allows data analysis with fuzzy data consolidation can be divided into two categories: data administration and data navigation.

First, an administrator of the OLAP system is enabled to define meta-objects used for data analysis later. These objects include cube definitions with their measures and dimensions, and contexts for dimension categorization. In our application, these contexts may be fuzzy, containing fuzzy equivalence classes as dimension categories.

Second, a business analyst (or any user of the system) is enabled to view, consolidate and navigate the decision support information. The user can view a cube, slice and dice dimensions, drill down or roll up consolidation levels of the data, change the order of dimensions, and change the aggregation function for data consolidation.

There are many commercial products meeting all of these requirements, with one exception: The consolidation of data using *fuzzy contexts*. This prototype is a first evaluation of the feasibility of fuzzy dimension categories in an OLAP-like data analysis application. This allows the navigation of fuzzy classifications of data in a graphical environment with some simple mouse-clicks.

### 8.1.2 System Architecture

The prototype has been implemented in the Java programming language. As illustrated by Figure 8.2, the system runs on a relational database server. The data basis as well as FC-OLAP metadata resides on this server.

FC-OLAP accesses the server via a JDBC driver. The graphical user interface has been programmed with Java/Swing. This GUI allows the definition of metadata objects, which are stored in the database server and reloaded every time a database connection is established. A fuzzy OLAP engine translates the user data administrations and manipulations into SQL queries. These SQL queries are transmitted to the database server via the JDBC driver.

### 8.1.3 Metadata

FC-OLAP requires the storage of metadata for the description of contexts and cubes with their dimensions. Furthermore, base relations are described together with their attributes and domains. Figure 8.3

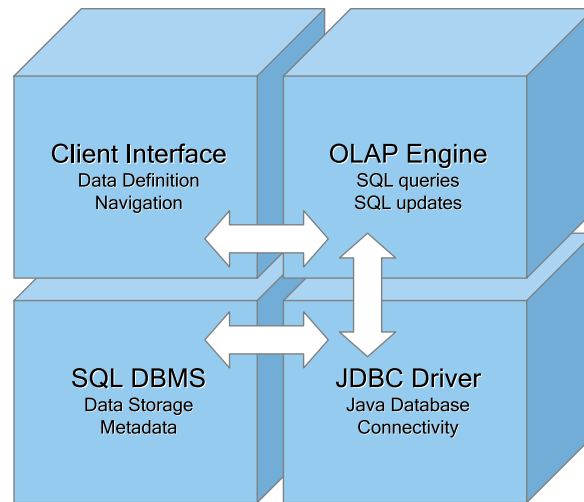


Figure 8.2: FC-OLAP System Architecture

shows an entity-relationship-like diagram of the metadata model implemented in FC-OLAP.

- Table `relations` contains a list of relations which are described in the metadata.
- Table `cubes` contains a list of cubes, their dimensions, their corresponding base relations, and their measure attribute.
- Table `dimensions` describes the dimensions of a cube and the underlying attribute.
- Table `attributes` contains the attributes and their data type for every relation.
- Table `elements` stores all elements of every attribute in all relations. In fact, this table would only be needed for attributes with discrete domains. *In a future implementation, discrete domains could be handled differently.*
- Table `contexts` contains the fuzzy and crisp contexts defined over an attribute. *In a future implementation, these contexts could be defined over an abstract domain instead of a specific attribute, such that the same context could be used in different cubes.*
- Table `classes` contains the classes for a context.
- Table `classMembership` stores the membership degree of a given set of values in a class. In the continuous case, these values in the `ValueList` present interpolation points for the calculation of membership degrees.
- Table `classification` contains for all elements in all attributes of all relations their degree of memberships in all context classes. Thus, this table is the most important table for the calculation of consolidated data cubes.

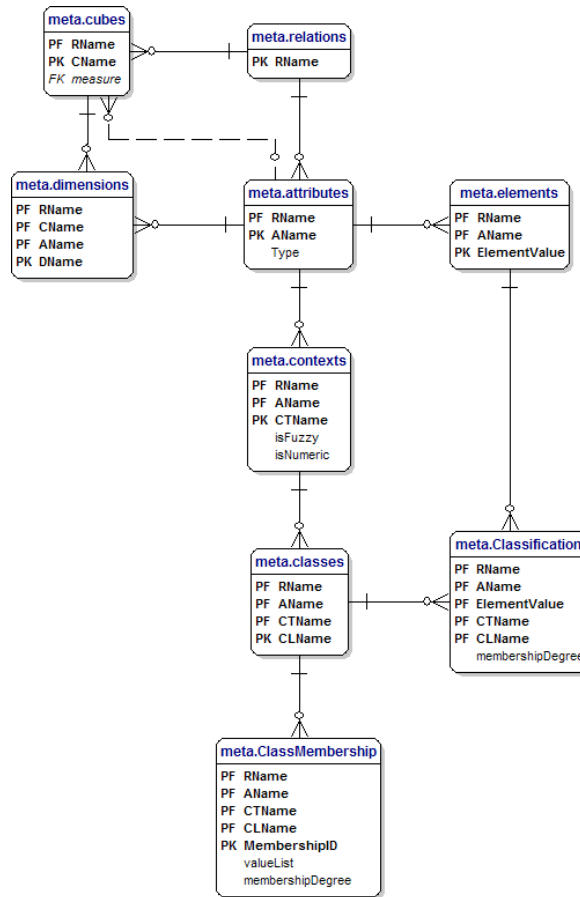


Figure 8.3: Metadata Structure Diagram

### 8.1.4 Implementation of Fuzzy Consolidation

In the crisp case, the data cell contents (the measure values belonging to a data cell) are calculated with a simple SQL statement.

```

select "<measure_attribute>" from "<relation>"
join meta.classification c_1
on ( c_1.rname = '<relation>'
and c_1.aname = '<dimension_1>'
and c_1.ctname = '<context_1>'
and c_1.elementvalue = "<dimension_1>" )
...
join meta.classification c_n
on ( c_n.rname = '<relation>'
and c_n.aname = '<dimension_n>'
and c_n.ctname = '<context_n>'

```

```

and c_n.elementvalue = "<dimension_n>")

where c_1.clname = <category_1>
...
and c_n.clname = <category_n>

```

The following example queries the content of the data cell containing the class of patients with a normal body mass index, a good cholesterol value, and a high blood pressure:

```

select "patient_id" from public."patient"
join meta.classification c_1
on ( c_1.rname = 'public."patient"'
and c_1.aname = 'bmi'
and c_1.ctname = 'level(bmi)'
and c_1.elementvalue = "bmi")
join meta.classification c1
on ( c_2.rname = 'public."patient"'
and c_2.aname = 'cstr1'
and c_2.ctname = 'level(cstr1)'
and c_2.elementvalue = "cstr1")
join meta.classification c2
on ( c_3.rname = 'public."patient"'
and c_3.aname = 'd_bp'
and c_3.ctname = 'level(d_bp)'
and c_3.elementvalue = "d_bp")
where c_1.clname = 'high'
and c_2.clname = 'bad'
and c_3.clname = 'normal'

```

The aggregation of the data cell content is accomplished using standard sql aggregation operators in the query. For crisp measure attribute classification, this operator is *distinct*.

```

select <aggregation> ( <measure> ) from ( <data_cell_content> )

```

In the fuzzy case, only the membership degrees for the attribute values in the dimension contexts are queried from the database. The classification of tuples and the aggregation of measure values are calculated in Java. The general form of a query selects for all tuples the membership degree in  $n$  contexts:

```

select "<measure_attribute>",
c_0.membershipdegree as m_1,
... ,
c_n.membershipdegree as m_n
from "<relation>"
join meta.classification c_1
on ( c_1.rname = '<relation>'
and c_1.aname = '<dimension_1>'
and c_1.ctname = '<context_1>'

```

```

and c_1.elementvalue = "<dimension_1>")
...
join meta.classification c_n
on ( c_n.rname = '<relation>'
and c_n.aname = '<dimension_n>'
and c_n.ctname = '<context_n>'
and c_n.elementvalue = "<dimension_n>")

```

The following example returns, for every tuple in the base relation, the measure attribute value `patient_id`, together with the membership degrees for all dimension attribute values in their dimension context.

```

select "patient_id",
c_1.membershipdegree as m_1,
c_2.membershipdegree as m_2,
c_3.membershipdegree as m_3
from public."patient"
join meta.classification c_1
on ( c_1.rname = 'public."patient"'
and c_1.aname = 'bmi'
and c_1.ctname = 'f_level'
and c_1.elementvalue = "bmi")
join meta.classification c1
on ( c_2.rname = 'public."patient"'
and c_2.aname = 'cstr1'
and c_2.ctname = 'f_level(cstr1)'
and c_2.elementvalue = "cstr1")
join meta.classification c2
on ( c_3.rname = 'public."patient"'
and c_3.aname = 'd_bp'
and c_3.ctname = 'f_level(d_bp)'
and c_3.elementvalue = "d_bp")
where c_1.clname = 'normal'
and c_2.clname = 'bad'
and c_3.clname = 'normal'

```

In the prototype application, the cube views are calculated by direct queries to the base relation on database server. This implementation is not scalable for large scale data analysis. In a future implementation, the cube contents should be stored in a star-scheme-like structure, such that the facts for every data cell could be queried directly, without calculation. This would greatly increase the computation time for cube definitions, but it would optimize access time for data analysis.

## 8.2 User Manual

### 8.2.1 System Installation

In order to run the FC-OLAP prototype, a PostgreSQL 8.0 database server must be up and running. Furthermore, the Java runtime environment must be installed on the system. If those conditions are

fulfilled, the installation of FC-OLAP consists in extracting the archive fcolap.zip into a folder of choice. In order to run FC-OLAP, type

```
java -jar fcolap.jar
```

In many graphical environments, double-clicking the icon will also start the application.

### 8.2.2 Data loading

All data for analysis must be loaded onto the database server manually. Specifically, the database administrator must create a relation in the database with the corresponding attributes. Then, the values can be read into the database using PostgreSQL's COPY command.

The relations used with FC-OLAP must meet certain requirements. First of all, the attributes of relations in the database server are supposed to contain orthogonal attributes. That is, their attributes should be functionally independent from each other. Second, the only data types the FC-OLAP prototype can handle so far are int2, int4, int8, float4, float8, text and varchar. Third, FC-OLAP does not accept NULL values in the relation.

If the data is entered in form of relations with orthogonal attributes containing compatible data types, and if the tuples in the relation do not contain NULL values, then FC-OLAP can calculate fuzzy OLAP cubes. The process of creating thus refined tables must be applied to raw data in advance. The aim of this process is to have a relation containing truly multidimensional data, where every tuple in the relation corresponds to a point in an information space.

### 8.2.3 Overview

After the FC-OLAP application has been started, the user interface is displayed. The general appearance of the user interface is shown in Figure 8.4.

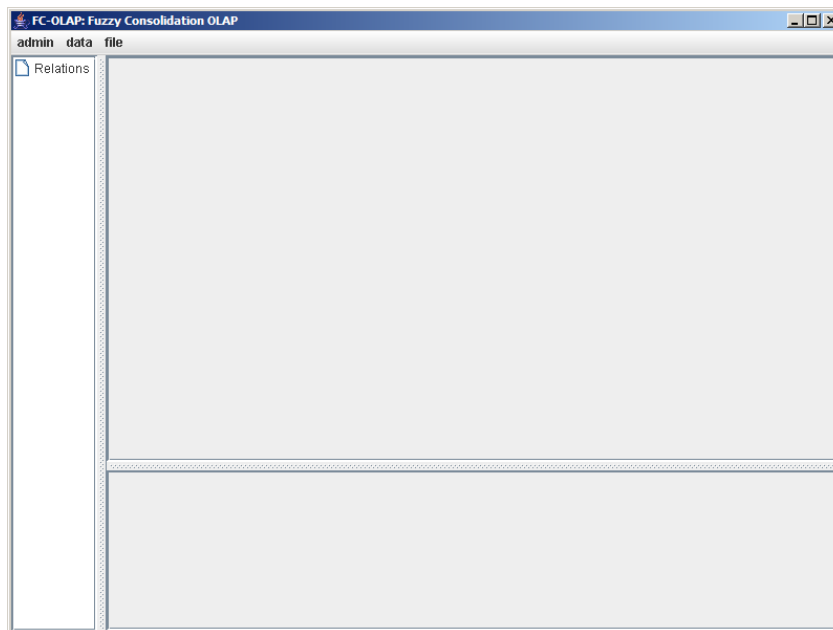


Figure 8.4: *FC-OLAP User Interface Look and Feel*

The interface consists of four parts: A menu bar, a meta object tree area on the left, the cube view area on the right, and a metadata panel on the bottom.

- The menu bar is used for all user inputs and controls. All actions of FC-OLAP are accessible via menu entries in the menu.
- The meta object tree shows all FC-OLAP meta objects defined in a database, such as cubes, dimensions, or contexts.
- The cube view area displays cube reports, that is, the facts for all data cells in a cube definition.
- The metadata panel displays information about the current cube view, such as the current cube, its dimensions and measure, the aggregation function, categorization levels and slice or dice conditions.

### 8.2.4 Data Administration

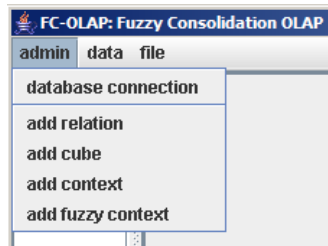


Figure 8.5: *The Admin Menu*

All user actions related to data administration are collected in the `admin` menu. Figure 8.5 shows the content of that menu. This menu allows establishing a database connection, adding relations to the metadata, and defining cubes, contexts and fuzzy contexts.

- *Establishing a Database Connection*

After clicking on the menu entry `admin / database connection`, a dialog appears, as shown by Figure 8.6.

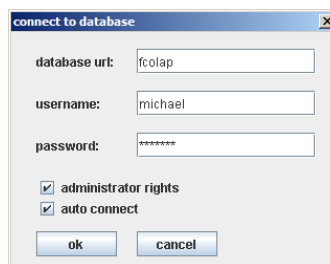


Figure 8.6: *Establishing a Database Connection*

Enter the name of the database server, and optionally its network domain and port number. Enter the user name and the corresponding password. If you want to change data in the database,

check the administrator rights checkmark. If you want to let FC-OLAP connect to this database automatically on startup, select the autoconnect checkmark.

- *Adding a Relation Description to the Metadata*

In order to define cubes over a relation, this relation must be added to the metadata manually.

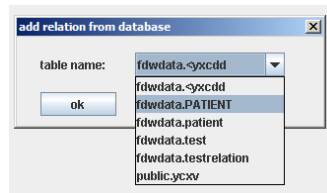


Figure 8.7: *Adding A Relation*

Clicking `admin / add relation` opens the dialog shown by Figure 8.7. This dialog lets the user choose a relation which resides on the database server. After clicking `ok` a description of the relation, its attributes and elements, is added to the metadata.

- *Defining a Cube*

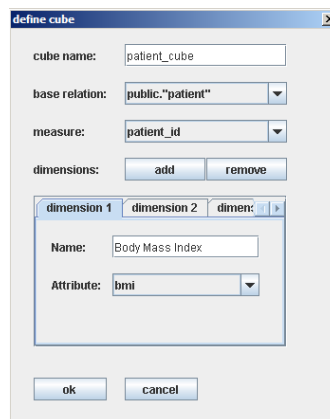


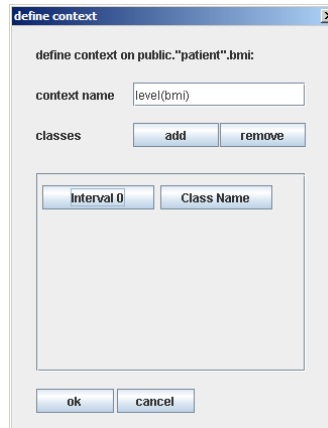
Figure 8.8: *Defining a Cube*

In order to define a cube, click `admin / add cube`. The dialog shown by Figure 8.8 appears. Choose the relation on which the cube is defined, a cube name, and the cube's measure attribute. Define one or more dimensions by giving a name and a base attribute.

- *Defining a Context*

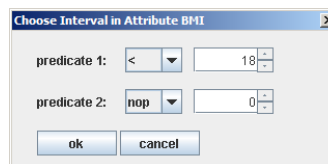
The menu entry `admin / define context` enables the user to define crisp contexts on attributes. In the prototype, context definitions are only possible on specific attributes of an existing relation. After choosing the relation and attributes, the context definition dialog (Figure 8.9) accepts context class definitions.

The `add` button next to the label `classes` adds a context class to the context definition. After clicking this button, two new buttons appear on the panel. The button labeled `interval` (or

Figure 8.9: *Defining a Context*

subdomain in the discrete case) allows the definition of elements which belong to the context class. The button labeled `class name` accepts the definition of a context class name.

Define the interval or subdomain for the context Class. If the domain is discrete, you can click on the elements that are in the class. If the domain is continuous, you can define an interval whose elements belong to the class in the dialog shown by Figure 8.10.

Figure 8.10: *Defining an Interval for a Context Class*

This dialog allows you to define two predicates in order to restrict the attribute domain to an interval subdomain. The predicates consist of a binary comparison operator and a value. For example, the operator `>=` and the value 3 define a predicate *x is greater than or equal to 3*.

The interval consists of elements satisfying both predicates. If you want to define only one predicate (for example, the interval of elements less than 18) then the second predicate can be disabled by choosing the `nop` operator.

- *Defining a Fuzzy Context*

The menu entry `admin / define fuzzy context` enables the user to define fuzzy contexts on attributes. After choosing the relation and attributes, the context definition dialog (Figure 8.11) accepts fuzzy context class definitions.

The fuzzy context definition panel allows the definition of fuzzy contexts in the following way. For every fuzzy context class, click on the `add` button next to the label `Fuzzy Classes`. For every class, you can choose a name by clicking on the `Class Name` button.

If the attribute's domain is continuous, the membership function is defined by interpolation points. An interpolation point is a value/membership degree pair. For a given value, you can choose an

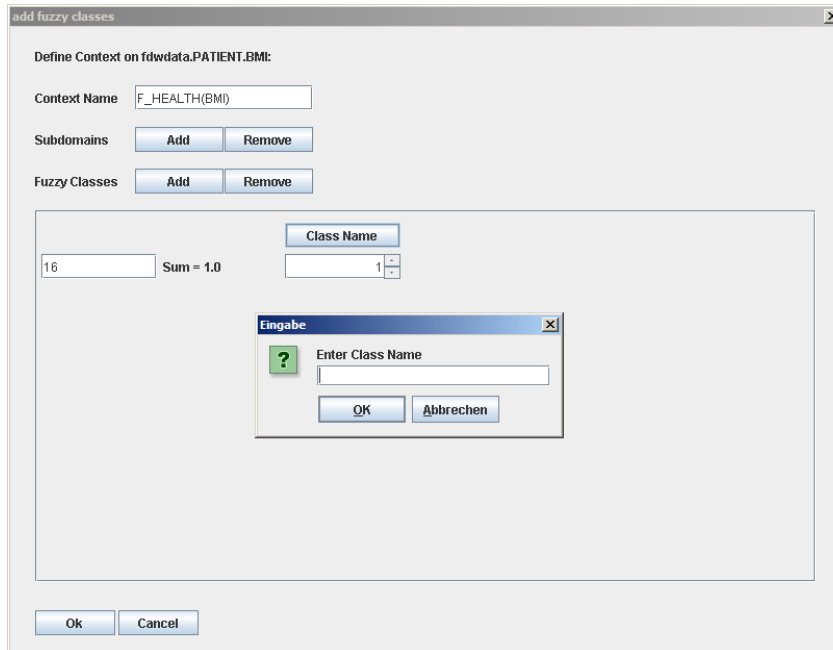


Figure 8.11: Defining a Fuzzy Context Class

interpolation membership degree for every class.

		BMI_UNHEA...	BMI_HEALT...
16	Sum = 1.0	1	0
20	Sum = 1.0	0	1
23	Sum = 1.0	0	1
27	Sum = 1.0	1	0

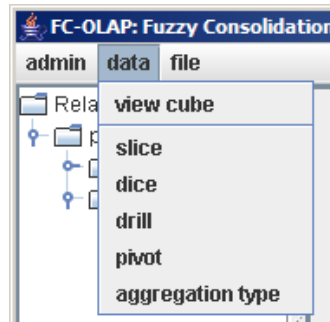
Figure 8.12: Defining a Fuzzy Context Class2

For example, the interpolation points that have been entered in the example shown by Figure 8.12 defines the fuzzy context  $f_{health}$ , which has been discussed in Example 3 in Section 6.4. Based on these interpolations, FC-OLAP calculates the membership function shown in Figure 6.3.

If the attribute's domain is discrete, you have to specify a membership degree for every element in the domain manually. For one or more elements you can specify a subdomain by clicking on the `subdomain` button and choosing the elements by checkmarking them. For all subdomains, choose a membership degree for every fuzzy class.

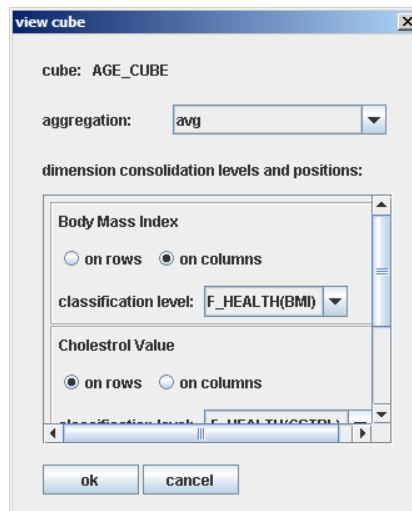
## 8.2.5 Data Navigation

The data menu (Figure 8.13) contains all user actions concerning data navigation and manipulation. This menu is accessed for viewing and navigating a cube. The possible navigations are slicing, dicing, drill-down/roll-up, pivoting, and changing the aggregation function of the cube.

Figure 8.13: *The Data Menu*

- *Creating a Cube View*

In order to display a cube to the screen, click `data / view cube`. Then, choose the cube to be displayed. After that, a dialog (Figure 8.14) appears, which expects the definition of the cube view.

Figure 8.14: *Creating a Cube View*

First, choose an aggregation function. For numeric measure attributes, this aggregation function can be one of the following:

- *Count*: Counting the number of tuples in the data cells
- *Sum*: Adding up the measure attribute values in the data cells
- *Avg*: Calculating the average of measure attribute values in the data cells
- *Min*: Displaying the minimal measure attribute value in the data cells
- *Max*: Displaying the maximal measure attribute value in the data cells
- *Classify*: Creating the class of measure values in the data cells

For categoric measure values, this aggregation function is one of the following:

- *Count*: Counting the number of measure values in the data cells
- *Classify*: Creating the class of measure values in the data cells

After that, for every dimension, you have to choose the consolidation level. This level consists of a crisp or fuzzy context defined on the dimensional attribute. Furthermore, for every dimension, choose its placement in the two-dimensional cube projection, which is displayed in the user interface. This placement is either on `rows` (that is, the dimension values are displayed in the row header), or on `columns` (the dimension values are displayed in the column header).

		Body Mass Index:			
		low	normal	high	All Columns:
Cholesterol Value:	Blood Pressure:				
good	normal	(A, 0.339) (B, 0.345) (E, 0.308) (F, 0.431) (G, 0.014) (I, 0.0060) (K, 0.053) (M, 0.0040)	(A, 0.312) (B, 0.449) (E, 0.013) (F, 0.353) (I, 0.067) (K, 0.188) (L, 0.278) (M, 0.0020) (N, 0.264) (V, 0.056) (W, 0.05)	(L, 0.031) (N, 0.083) (P, 0.126) (V, 0.197) (W, 0.2)	(A, 0.651) (B, 0.794) (E, 0.321) (F, 0.784) (G, 0.014) (I, 0.073) (K, 0.241) (L, 0.309) (M, 0.0050) (N, 0.347) (P, 0.126) (V, 0.253) (W, 0.25)
good	high	(A, 0.159) (B, 0.02) (E, 0.631) (F, 0.0070)	(A, 0.187) (B, 0.023) (E, 0.026) (F, 0.0060)	(C, 0.725) (H, 0.058) (L, 0.022) (N, 0.073)	(A, 0.346) (B, 0.043) (C, 0.725) (E, 0.657)

cube: patient\_cube  
measure: patient\_id  
aggregation function: classify  
dimensions on columns: Body Mass Index  
dimensions on rows: Cholesterol Value, Blood Pressure  
consolidation: Body Mass Index -> f\_level  
consolidation: Cholesterol Value -> f\_level(cstr1)  
consolidation: Blood Pressure -> f\_level(d\_bp)

Figure 8.15: Resulting Cube View

Figure 8.15 shows an example of a resulting cube view on the example relation *patient* (Table 7.1), with *classify* as aggregation function. The dimensions *Cholesterol Value* and *Blood Pressure* are displayed on rows, and the dimension *Body Mass Index* is displayed on columns. The data cells contain the fuzzy class of patients whose health indicators are classified by the data cell's dimension values.

- *Slicing A Cube*

Slicing a dimension of a cube means fixing it to a certain value. In order to slice a dimension, click `data / slice` and choose the dimension to be sliced. After that, choose the slice value (Figure 8.16).

Only data cells who contain the value `normal` for the dimension *Body Mass Index* are displayed in the resulting cube view.

For example, in the previous cube view, slicing the dimension *Body Mass Index* to the value `normal` results in a cube view illustrated by Figure 8.17.

- *Dicing a Cube*

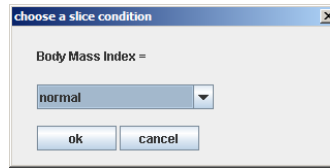


Figure 8.16: Slicing a Cube's Dimension

	Body Mass Index:	normal	All Columns:
Cholesterol Value:	Blood Pressure:		
good	normal	(A, 0.312) (B, 0.449) (E, 0.013) (F, 0.353) (I, 0.067) (K, 0.188) (L, 0.278) (M, 0.0020) (N, 0.264) (V, 0.056) (W, 0.05)	(A, 0.312) (B, 0.449) (E, 0.013) (F, 0.353) (I, 0.067) (K, 0.188) (L, 0.278) (M, 0.0020) (N, 0.264) (V, 0.056) (W, 0.05)
good	high	(A, 0.187) (B, 0.023) (E, 0.026) (F, 0.0060) (H, 0.67) (I, 0.041)	(A, 0.187) (B, 0.023) (E, 0.026) (F, 0.0060) (H, 0.67) (I, 0.041)

cube: patient\_cube  
 measure: patient\_id  
 aggregation function: classify  
 dimensions on columns: Body Mass Index  
 dimensions on rows: Cholesterol Value, Blood Pressure  
 consolidation: Body Mass Index -> f\_level(cstrf)  
 consolidation: Cholesterol Value -> f\_level(cstrf)  
 consolidation: Blood Pressure -> f\_level(d\_bp)  
 slice: Body Mass Index = normal

Figure 8.17: Cube Resulting From a Slice Operation

Dicing a cube's dimensions means restricting the possible dimension values. In order to dice a dimension, click *data / dice* and choose the dimension to be diced. After that, choose the domain restriction. If the dimension values are discrete, or if the dimension is consolidated by a context, you can choose the elements in the restricted domain individually by checkmarking them (Figure 8.18).

If the dimension's domain is continuous and the consolidation level is the basic granularity, you can choose an interval for the restricted domain. The interval definition is the same as the one used for continuous context class subdomains (see Figure 8.9)

Figure 8.19 shows an example of a domain restriction, where the dimension *Body Mass Index* has been restricted to the values *normal* and *high*.

- *Drilling the Consolidation Level of a Cube*

This operation is usually known as drill-down or roll-up. In FC-OLAP, a consolidation level change is accomplished by choosing a new context for a dimension. In order to roll up or drill down, click *data / drill*, choose the dimension, and then choose the new consolidation level by clicking on a context in the option dialog (Figure 8.20).

In the resulting cube view, a new set of data cells is displayed, one for each possible combination

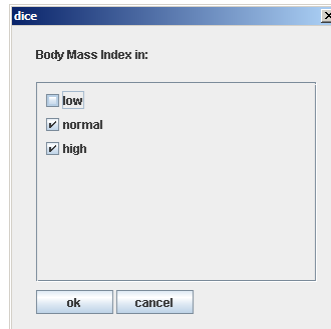


Figure 8.18: *Dicing a Cube*

of dimension values. The new dimension values for the drilled dimensions are the classes of the chosen context.

Consider the cube view shown in Figure 8.19. If we roll up the dimension `Body Mass Index` from context `f_level` to `f_health`, the resulting cube view is the one shown in Figure 8.21.

- *Pivoting*

The placement of two dimensions in the cube view can be switched by clicking `data / pivot`. Choose the two dimensions to be pivoted. After that, the resulting cube view places the first dimension on the place of the second, and vice versa.

## 8.2.6 Data Input and Output

The file menu (Figure 8.22) encapsulates actions concerning data input and output. It allows the user to export the data of the current cube view, to save the cube view specification to the disk, and to load a saved cube view specification from the disk.

- *Exporting Cube View Data*

click `file / export view content` in order to export a cube view to a table in CSV (Comma Separated Values) format. A file chooser dialog appears, where you can choose the filename and folder of the CSV output file.

- *Saving A Cube View Specification*

Clicking `file / save view config` allows the user to save the specification of the current cube view. This allows the user to recreate the same view later, without having to specify consolidation levels, slice and dice conditions and aggregation functions again. A file chooser dialog appears, where you can choose the filename and folder of the data output file.

- *Opening A Cube View Specification*

In order to load the specification of a cube view, click `file / open view config`. A file chooser dialog appears, where you can choose the file which contains the cube view specification.

The screenshot shows the FC-OLAP: Fuzzy Consolidation OLAP application window. On the left, there is a tree view with 'Relations' containing 'public."patient"' and 'Cubes' containing 'patient\_cube' and 'age\_cube'. The main area displays a table with columns: 'Cholesterol Value', 'Blood Pressure', 'Body Mass Index', and 'All Columns:'. The table shows data for 'good' cholesterol and 'normal' blood pressure, with 'Body Mass Index' values ranging from 'normal' to 'high'. Below the table, configuration details are listed: 'cube: patient\_cube', 'measure: patient\_id', 'aggregation function: classify', 'dimensions on columns: Body Mass Index', 'dimensions on rows: Cholesterol Value, Blood Pressure', 'consolidation: Body Mass Index -> f\_level', 'consolidation: Cholesterol Value -> f\_level(cstr1)', 'consolidation: Blood Pressure -> f\_level(d\_bp)', and 'dice: Body Mass Index in {normal, high}'.

Figure 8.19: Cube Resulting From a Dice Operation

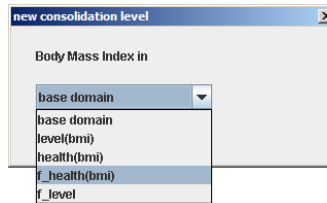


Figure 8.20: Changing a Cube Dimension's Consolidation Level

### 8.3 Evaluation and Outlook

Our software prototype is a part of a feasibility study. The aim of FC-OLAP is to show how fuzzy classification can be applied to OLAP cubes. This aim has been fulfilled. The examples discussed in the thesis have been successfully modeled and calculated using FC-OLAP. These examples and all necessary data can be found on the CD-ROM on which the FC-OLAP source is delivered. Thus, the implementation can be evaluated as sufficient to show the feasibility of the proposed framework for fuzzy data consolidation.

Another advantage of FC-OLAP is that it provides the functionality of fuzzy data classification in an easy-to-use graphical interface. An existing implementation of a fuzzy classification tool fCQL (Meier *et al.*, 2003) was based on written queries in the style of the SQL language. For users in a real world application, the graphical data manipulation is much easier to handle and faster to learn. In fact, FC-OLAP is capable of providing the same functionality of FCQL in a more intuitive user interface. The only difference is that in fCQL, the fuzzy classes are named, while in FC-OLAP, those classes are automatically generated by the distribution of dimension values.

Furthermore, what is new in FC-OLAP is the possibility to navigate fuzzy classes in OLAP-style,

Cholesterol Value:	Blood Pressure:	Body Mass Index:	healthy	unhealthy	All Columns:
good	normal	(A, 0.552) (B, 0.76) (E, 0.096) (F, 0.637) (I, 0.073) (K, 0.241) (L, 0.309) (M, 0.0030) (N, 0.33) (V, 0.069) (W, 0.063)	(A, 0.099) (B, 0.035) (E, 0.225) (F, 0.147) (G, 0.014) (M, 0.0020) (N, 0.017) (P, 0.126) (V, 0.183) (W, 0.188)	(A, 0.651) (B, 0.794) (E, 0.321) (F, 0.784) (G, 0.014) (I, 0.073) (K, 0.241) (L, 0.309) (M, 0.0050) (N, 0.347) (P, 0.126) (V, 0.253) (W, 0.25)	
good	high	(A, 0.32) (B, 0.04) (E, 0.197) (F, 0.01)	(A, 0.026) (B, 0.0030) (C, 0.725) (E, 0.46)	(A, 0.346) (B, 0.043) (C, 0.725) (E, 0.667)	

cube: patient\_cube  
measure: patient\_id  
aggregation function: classify  
dimensions on columns: Body Mass Index  
dimensions on rows: Cholesterol Value, Blood Pressure  
consolidation: Body Mass Index -> f\_health(bmi)  
consolidation: Cholesterol Value -> f\_level(cstrl)  
consolidation: Blood Pressure -> f\_level(d\_bp)

Figure 8.21: Cube Resulting From a Drill Operation

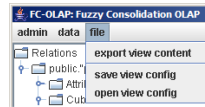


Figure 8.22: The File Menu

that is by slicing, dicing, drilling down and rolling up the dimensions of a multidimensional data cube. This combines the analytical approach of OLAP with fuzzy classification.

Still, FC-OLAP is only a prototype. Many features necessary for a functional application are lacking. This section gives some thought on the shortcomings of the FC-OLAP prototype, and proposes improvements for a possible future implementation, in order to provide a fully functional software tool for data analysis.

- *Scalability:* The prototype is not scalable for large amounts of data. This is because cube contents are not cached in a star scheme. This could be improved by precompiling the cube content when a cube is defined. A star scheme would contain the facts for every possible combination of dimension values, dimension consolidation levels, and aggregation functions. However, this star scheme would have to be extended to fuzzy dimension categories. A fuzzy dimension table could store for every dimension value the categories together with the membership degree as a pair. A different approach would be to extend the dimension tables, where the rows correspond to category names, and the cells store only a numeric membership degree. Anyway, the development of a *fuzzy star scheme* is one of the great challenges of a future implementation of fuzzy OLAP.
- *Usability:* The prototype's usability is restricted, because it only allows the definition of meta

objects, but not their removal or deletion. A future implementation should provide the possibility of editing and deleting relations, cubes, dimensions, contexts, and classes.

- *Security:* The current implementation is not secure, because the database server password must be known in order to connect to the database server. A future implementation could provide a three-tiered architecture using a web application server and a web user interface. Thus, the password would securely reside on the web server, and the users should access the interface using a standard web browser.
- *Definition of contexts on other contexts:* In the prototype, it is only possible to define contexts directly on the attribute's domain. However, in order to define dimension category hierarchies, it would be comfortable to be able to define contexts on top of existing contexts. Furthermore, in the crisp case, this also saves calculation time.
- *Reusable Contexts:* In the prototype implementation, contexts can only be defined on specific attributes. Therefore, contexts that have been defined once can not be used for different relations. The solution is the definition of contexts on abstract domains. Then, relational attributes are associated with abstract domains, and thus, the contexts defined on the domain are applicable to the attribute. Using this approach, contexts defined once can be used for an arbitrary number of relational attributes with a compatible domain.
- *Predefined Time Categories:* Contexts on time attributes are always the same, namely hours, days, months, years, et cetera. Thus, these time-related contexts could be predefined in the application. This would facilitate the historization aspect of a data warehouse.
- *Data Types:* The range of data types which can be handled should be extended in a future version.
- *Charting:* For crisp data cubes, a future implementation could provide graphical support for the display of charts visualizing the cube's contents.

# Chapter 9

## Conclusion

### 9.1 Summary of Results

The question about the use of fuzzy classification in a data warehouse is answered by a concept of OLAP-like data analysis using fuzzy classes for the consolidation of raw data into decision support information. The development of the mathematical framework and the implementation of the prototype application present a feasibility study for fuzzy classification in OLAP.

The concept of fuzzy data consolidation has been inspired by multidimensional data analysis described in (Codd, 1993) and the concept of fuzzy aggregation presented in (Schepperle *et al.*, 2004). The main approach is to turn the abstract idea into a specific mathematical framework and a corresponding prototype computer program.

The basic idea of fuzzy data consolidation is that an OLAP-cube presents a collection of multidimensional classes of data elements. Every data cell corresponds to a class whose classification predicates are given by the dimension values. Accordingly, if a fuzzy OLAP cube contains fuzzy dimension categories, its data cells correspond to fuzzy classes defined by the different membership degrees of the dimension values of a data element in those categories.

The mathematical framework consists of five parts: (1) multidimensional information, (2) fuzzy attribute categorization, (3) fuzzy tuple classification, (4) aggregation of fuzzy classes and finally (5) the layout of fuzzy-aggregated information in OLAP-cubes. First, every Cartesian product of linearly independent data domains represents a multidimensional point lattice, which is a kind of discrete counterpart to a vector space. The attribute's domains present dimensions, and tuples present points in that space. Second, fuzzy contexts can be used for dimension categorization in OLAP cubes. Third, the combination of classification features using fuzzy intersection leads to fuzzy tuple classification. Using the algebraic product operator provides summarizable data aggregation. Fourth, tuples in a fuzzy class can be aggregated by fuzzified relational aggregation operators such as sum, count, average or minimum / maximum. Fifth, this information can be represented by OLAP-cubes, in a way that allows the navigation of fuzzy classes by slicing, dicing, drill-down and roll-up operators. The drill-down and roll-up of consolidation levels can be accomplished by applying different fuzzy contexts on the dimensional attributes.

The advantage of fuzzy classification is a more accurate allocation of tuples to classes. Examples 16 and 20 illustrate this point. Relation *patients* (Table 7.1) contains information about two patients with an identification  $V$  and  $W$ . Both patients have the same health indicator values, with a difference of some decimal points. But because this negligibly small difference is exactly at the limit of the crisp classes, patient  $V$  is classified in the most healthy class, but patient  $W$  is classified in the most endangered class

(see Example 16). Using fuzzy classification removes this distortion, and the two patients are assigned to multiple classes to certain degrees (see Example 20).

A software prototype has been developed, which implements an approach to data analysis based on fuzzy classification in combination with OLAP cubes. This prototype allows the definition of OLAP cubes with fuzzy dimension categories. These cubes can be viewed and navigated using the OLAP operators slice, dice, and drill, which have been extended to fuzzy dimension categorizations. Such a data analysis tool provides an intuitive access to the navigation and consolidation of fuzzy-classified information.

## 9.2 Personal Evaluation

Joseph Weizenbaum, Professor emeritus of MIT and inventor of the famous ELIZA program (Weizenbaum, 1966), wrote the epilogue to the book (Jähn & Nagel, 2003). He writes “*One began thus with the search for an intended purpose for an existing technological development. However, we should ask questions about the actual need, instead of beginning with a coincidentally feasible solution.*” When a new technology is invented, sometimes what is missing is a problem for the solution. Even (Codd, 1993) admits that “*IT should never forget that technology is a means to an end, not an end in itself.*” The same holds true for the application of data warehousing in healthcare.

Fuzzy classification and data warehousing are existing technologies. Searching for an intended purpose of those technologies in the field of e-Health without a recognizable *need* is exactly the mistake professor Weizenbaum warned about in the epilogue of (Jähn & Nagel, 2003). I believe it is wrong to impose an existing technology on a field of application without obvious requirement, especially when it comes to sensitive data in the area of healthcare.

A possible application of OLAP with fuzzy data consolidation is a data warehouse of patient medical records in a hospital. Data cells would represent different classes of patients which imply different patient treatments concerning medication, diet and therapy. Another application in a health insurance company would be to classify their customers according to their electronic medical record. Different classes of customers could be provided with specialized prevention or therapy information according to fuzzy categories of their attributes. Such an information system would certainly combine the technologies of fuzzy classification, data warehousing and e-Health. However, the question remains whether that system is actually *needed*.

The presented framework for fuzzy classification in OLAP might be applied to customer relationship management in general. For example, many companies have an interest in classifying their customers according to certain features. This classification can be described by linguistic variables or fuzzy contexts. A software using the concepts presented by our FC-OLAP prototype would allow fuzzy-classification-based data analysis in an intuitive, graphical software environment.

## 9.3 Outlook

The mathematical framework presented in Chapter 7 is a first attempt in modeling multidimensional data hypercubes with fuzzy dimension categorization. Anyway, especially in mathematics and informatics, the statement *small is beautiful* is very true. Accordingly, a future challenge might be the development of a simpler conceptual model without loss of expression power.

The software prototype showed that fuzzy classification can be applied to OLAP cubes. However, during the process of implementation, two conclusions arose for the future pursuit of fuzzy classification in OLAP. First, the fuzzification of dimensional categories leads to a dimensionality problem. Using

---

fuzzy classification, every data element has to be classified in every class of the corresponding categorization level for every dimension. This increases the calculation time to the power of the number of dimension categories in comparison to the crisp case. This has to be considered when a scalable application is built. Second, a fuzzy class is not as easily understandable as a crisp class. Non-technical users may not be familiar with the concept of numerical membership degrees. Thus, a real-world application might defuzzify a fuzzy class to make the resulting cubes more readable.

As discussed in Section 8.3, the software developed in this master's thesis is far from being operational. Many features are lacking which are necessary for a scalable deployment of an OLAP system in the real world. Yet, the FC-OLAP prototype presents an extensible framework for the application of fuzzy classification to OLAP data analysis. If the approach of fuzzy data consolidation turns out to be worth a further research effort, a future implementation might reuse parts of the FC-OLAP program. The object-oriented metadata model, the JDBC database access and the model-view-controller architecture of the graphical user interface present reusable templates upon which, without starting from scratch a more scalable OLAP tool for fuzzy classification can be built.

# References

- Agrawal, Rakesh, Gupta, A., & Sarawagi, Sunita. 1997. Modeling Multidimensional Databases. *Pages 232–243 of: Gray, Alex, & Larson, Per-Åke (eds), Proc. 13th Int. Conf. Data Engineering, ICDE.* IEEE Computer Society.
- Apermann, Jens. 2003. Online Apotheken und e-Rezepte. *In: Jähn, Karl, & Nagel, Eckhard (eds), e-Health.* Springer.
- Bandemer, Hans, & Gottwald, Siegfried. 1993. *Einführung in Fuzzy Methoden: Theorie und Anwendungen.* Akademie Verlag.
- Beauchamp, T. L., & Childress, J. F. 2001. *Principles of biomedical ethics.* fifth edn. Oxford University Press.
- Borchers, Detlef. 2004. *Elektronische Gesundheitskarten auf dem Weg.* <http://www.heise.de/newsticker/meldung/56328>, Last accessed: 27.08.2005.
- Claus, Volker, & Schwill, Andreas (eds). 2001. *Duden Informatik.* Third edn. Dudenverlag.
- Codd, E. F. 1970. A relational model of data for large shared data banks. *Commun. ACM*, **26**(1), 64–69.
- Codd, E.F. 1993. *Providing OLAP (On-Line Analytical Processing) to User-Analysts - An IT Mandate.* a Hyperion Solutions Whitepaper.
- Colossi, N., Malloy, M., & Reinwald, B. 2002. Relational extensions for OLAP. *IBM Systems Journal*, **41**(4), 714–731.
- Davis, Ernest. 1990. *Representations of Commonsense Knowledge.* Morgan Kaufmann Publishers.
- Dietzel, Gottfried T. W. 2003a. Auf dem Weg zur europäischen Gesundheitskarte und zum e-Rezept. *In: Jähn, Karl, & Nagel, Eckhard (eds), e-Health.* Springer.
- Dietzel, Gottfried T. W. 2003b. Teleservices in der Praxis. *In: Jähn, Karl, & Nagel, Eckhard (eds), e-Health.* Springer.
- Einbinder, Jonathan S., Scully, Kenneth W., Pates, Robert. D., Schubart, Jane R., & Reynolds, Robert E. 2001. Case Study: A Data Warehouse for an Academic Medical Center. *Journal of Healthcare Information Management*, **15**(2), 165–175.
- Eysenck, Michael W., & Keane, Mark T. 2000. *Cognitive Psychology.* Fourth edn. Psychology Press.
- Gramann, Klaus Dieter Meyer. 1994. Fuzzy Classification: An Overview. *In: Kruse, Rudolf, Gebhardt, Jörg, & Palm, Rainer (eds), Fuzzy Systems in Computer Science.* Vieweg.

- Gray, Jim, Chaudhuri, Surajit, Bosworth, Adam, Layman, Andrew, Reichart, Don, Venkatrao, Murali, Pellow, Frank, & Pirahesh, Hamid. 1997. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *J. Data Mining and Knowledge Discovery*, **1**(1), 29–53.
- Inmon, W. H. 1996. *Building the Data Warehouse*. Second edn. John Wiley and Sons.
- Jähn, Karl, & Nagel, Eckhard (eds). 2003. *e-Health*. Springer.
- JSR. 2003. *JOLAP - Java OLAP Interface Specification, Proposed Final Draft*.
- Kriegel, Alex, & Trukhnov, Boris M. 2003. *SQL Bible*. John Wiley and Sons.
- Krohs, Ulrich. 2003. Angewandte Ethik e-Health. In: Jähn, Karl, & Nagel, Eckhard (eds), *e-Health*. Springer.
- Laurent, Anne. 2002. *Base de données multidimensionnelles floues et leur utilisation pour la fouille de données*. Ph.D. thesis, Université Paris 6, Paris, France.
- Laurent, Anne. 2003. Querying fuzzy multidimensional databases: unary operators and their properties. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, **11**(Supplement), 31–45.
- Ledbetter, Craig S., & Morgan, Matthew W. 2001. Toward best practice: leveraging the electronic patient record as a clinical data warehouse. *Journal of Healthcare Information Management*, **15**(2), 119–131.
- Lehner, Wolfgang. 2003. *Datenbanktechnologie für Data-Warehouse-Systeme*. Dpunkt Verlag.
- Markus T.J.Mohr, Thomas Shcall und Michael Nerlich. 2003. Telemedizin. In: Jähn, Karl, & Nagel, Eckhard (eds), *e-Health*. Springer.
- Meier, Andreas. 2001. *Relationale Datenbanken*. Springer Verlag.
- Meier, Andreas, Mezger, Christian, Werro, Nicolas, & Schindler, Günter. 2003. Zur unscharfen Klassifikation von Datenbanken mit fCQL. *Pages 151–158 of: Bauer, Andreas, Böhnlein, Michael, Herden, Olaf, & Lehner, Wolfgang (eds), Proceedings of the GI-Workshop LLWA Lernen, Lernen, Wissen, Adaptivität*. Shaker.
- Messerschmidt, Hartmut, & Schweinsberg, Kai. 2003. *OLAP mit dem SQL-Server*. Dpunkt Verlag.
- Moser, Paul K., Mulder, Dwayne H., & Trout, J.D. 1993. *Theory of Knowledge*. Oxford University Press.
- Nguyen, Hung T., & Walker, Elbert A. 1997. *A First Course in Fuzzy Logic*. CRC Press.
- OMG. 2003. *CWM - Common Warehouse Metamodel Specification, Version 1.1*. Online: <http://www.omg.org/cgi-bin/apps/doc?formal/03-03-02.pdf>, Last accessed: 27.08.2005.
- Pirrotte, Alain. 1982. A Precise Definition of Basic Relational Notions and of the Relational Algebra. *ACM SIGMOD Record*, **13**(1), 30–45.
- Ramming, Joachim. 2003a. Integrierte Gesundheitsversorgung. In: Jähn, Karl, & Nagel, Eckhard (eds), *e-Health*. Springer.

- Ramming, Joachim. 2003b. Integrierte Gesundheitsversorgung. *In: Jähn, Karl, & Nagel, Eckhard (eds), e-Health*. Springer.
- Rundensteiner, Elke, & Bic, Lubomir. 1992. Evaluating Aggregates in Possibilistic Relational Databases. *Data & Knowledge Engineering*, **7**, 239 – 267.
- Schepperle, Heiko, Merkel, Andreas, & Haag, Alexander. 2004. Erhalt von Imperfektion in einem Data Warehouse. *Pages 33–42 of: Bauer, Andreas, Böhnlein, Michael, Herden, Olaf, & Lehner, Wolfgang (eds), Internationales Symposium: Data-Warehouse-Systeme und Knowledge-Discovery*. Shaker.
- Schilp, Jill Lenk, & Gilbreath, Roy E. (eds). 2000. *Health Data Quest: How to Find and Use Data for Performance Improvement*. San Francisco: Jossey-Bass Inc.
- Schindler, Günter. 1998. *Fuzzy-Datenanalyse Durch Kontextbasierte Datenbankanfragen*. Ph.D. thesis, Technische Hochschule Aachen.
- Schlegel, Thomas. 2003. Rechtsfragen im Überblick. *In: Jähn, Karl, & Nagel, Eckhard (eds), e-Health*. Springer.
- Schramm-Wölk, Ingeborg, & Schug, Stephan H. 2003. e-Patientenakte und e-Gesundheitsakte. *In: Jähn, Karl, & Nagel, Eckhard (eds), e-Health*. Springer.
- Shenoi, Sujeet. 1995. Fuzzy Sets, Information Clouding, and Database Security. *In: Bosc, P., & Kacprzyk, J. (eds), Fuzziness in Database Management Systems*. Physika Verlag, Heidelberg.
- Skopal, Tomas, Kratky, Michal, Snasel, Vaclav, & Pokorny, Jaroslav. 2003. *On Range Queries in Universal B-trees*. Tech. rept. Amphora Research Group.
- Vassiliadis, Panos, & Sellis, Timos K. 1999. A Survey of Logical Models for OLAP Databases. *SIGMOD Record*, **28**(4), 64–69.
- Warda, Frank, & Noelle, Guido. 2002. *Telemedizin und eHealth in Deutschland: Materialien und Empfehlungen für eine nationale Telematikplattform*. Deutsches Institut für medizinische Dokumentation und Information.
- Weisstein, Eric W. 2005. *Point Lattice*. From MathWorld—A Wolfram Web Resource. <http://mathworld.wolfram.com/PointLattice.html>, Last accessed: 27.08.2005.
- Weizenbaum, Joseph. 1966. ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine. *Communications of the Association for Computing Machinery*, **9**, 36–45.
- Zadeh, L. A. 1965. Fuzzy Sets. *Information and Control*, **8**, 338–353.
- Zadeh, L. A. 1973. *The Concept Of A Linguistic Variable And Its Applications To Approximate Reasoning*. Tech. rept. Memorandum ERL-M 411 Berkeley.
- Zadeh, Lofti A. 1975. Calculus of fuzzy restrictions. *In: Zadeh, Lofti A., Fu, King-Sun, Tanaka, Kokichi, & Shimura, Masamichi (eds), Fuzzy Sets and their Applications to Cognitive and Decision Process*. Academic Press.
- Zimmermann, Hans-Jürgen. 1993. *Fuzzy Set Theory and its Applications*. Second edn. Kluwer Academic Publishing.