# Data-Centric Systems and Applications

Carlo Batini · Monica Scannapieca

# Data Quality

## Concepts, Methodologies and Techniques

With 134 Figures

Springer

*Carlo Batini*

Università di Milano Bicocca
Dipartimento di Informatica
Sistemistica e Comunicazione
Piazza dell'Ateneo Nuovo
20126 Milano
Italy
batini@disco.unimib.it

*Monica Scannapieca*

Dipartimento di Informatica e Sistemistica "A. Ruberti"
Università di Roma "La Sapienza"
Via Salaria 113
00198 Roma
Italy
monscan@dis.uniroma1.it

Library of Congress Control Number: 2006931550

ACM Computing Classification (1998): H.2, H.1

ISBN-10 3-540-33172-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-33172-8 Springer Berlin Heidelberg New York



Cover Design: KünkelLopka, Heidelberg
Typesetting: by the Authors
Production: LE-TEX Jelonek, Schmidt & Vöckler GbR, Leipzig

Printed on acid-free paper 45/3100/YL 5 4 3 2 1 0

*To my wonderful children, Chiara, Laura, and Giulio.*
Carlo

*To Massimo and to my "Ernania" world.*
Monica

# Preface

## Motivation for the Book

Electronic data play a crucial role in the information and communication technology (ICT) society: they are managed by business and governmental applications, by all kinds of applications on the Web, and are fundamental in all relationships between governments, businesses, and citizens. Because electronic data is so widely diffused, the "quality" of such data and its related effects on every kind of activity of the ICT society are more and more critical.

The relevance of data quality in both decisional and operational processes is recognized by several international institutions and organizations. As an example, the importance of data quality in decisional processes is clearly stated in the quality declaration of the European Statistical System [72], in which its mission is identified as follows: "We provide the European Union and the world with high quality information on the economy and society at the European, national, and regional levels and make the information available to everyone for decision-making purposes, research, and debate."

Furthermore, quality of data is also a significant issue for operational processes of businesses and organizations. The Data Warehousing Institute in a 2002 report on data quality (see [52]) shows that there is a significant gap between perception and reality regarding the quality of data in many organizations, and that data quality problems cost U.S. businesses more than 600 billion dollars a year.

The "Year 2000 problem", which led to modify software applications and databases using a two-digit field to represent years, has been a data quality problem. The costs to modify such software applications and databases have been estimated to be around 1.5 trillion US dollars (see [68]).

Some disasters are due to the presence of data quality problems, among them the use of inaccurate, incomplete, out-of-date data. For example, the ex-

plosion of the space shuttle Challenger is discussed in [78] according to a data quality perspective; the analysis reports more than ten different categories of data quality problems having a role in the disaster.

Such errors are motivations at the basis of the several initiatives that are being launched in the public and private sectors, with data quality having a leading role, as detailed in Chapter 1; the initiatives include, for instance, the Data Quality Act effected by the United States government in 2002 [149].

Electronic data are only to a certain extent of better quality than data stored in paper documents. Indeed, electronic data benefit from a defined and regulated representation, but processes that originate such data are often out of control, and consequently errors in data proliferate.

In the last decades, information systems have been migrating from a hierarchical/monolithic to a network-based structure, where the potential sources that organizations can use for the purpose of their businesses is dramatically increased in size and scope. Data quality problems have been further worsened by this evolution. In networked information systems, processes are involved in complex information exchanges and often operate on input obtained from other external sources, frequently unknown a priori.

As a consequence, the overall quality of the information that flows between information systems may rapidly degrade over time if both processes and their inputs are not themselves subject to quality control. On the other hand, the same networked information system offers new opportunities for data quality management, including the possibility of selecting sources with better quality data, and of comparing sources for the purpose of error localization and correction, thus facilitating the control and improvement of data quality in the system.

Due to the described above motivations, researchers and organizations more and more need to understand and solve data quality problems, and thus need answering the following questions: What is, in essence, data quality? Which techniques, methodologies, and data quality issues are at a consolidated stage? Which are the well-known and reliable approaches? Which problems are open? This book is an attempt to respond to all these questions.

## Goals

The goal of this book is to provide a systematic and comparative description of the vast number of research issues related to quality of data, and thus to illustrate the state of the art in the area of data quality. While being a real problem in a vast number of activities in the private and public sectors, data quality recently resulted in a significant number of contributions to the research community. There are several international conferences promoted by the database and information system communities that have data quality as their main topic; the International Conference on Information Quality (ICIQ) [95], organized traditionally at the Massachusetts Institute of

Technology (MIT) in Boston, started in 1996; the International workshop on Information Quality in Information Systems (IQIS) [99], held in conjunction with the SIGMOD conference since 2004; the international workshop on Data and Information Quality (DIQ), held in conjunction with the Conference on Advanced Information Systems Engineering (CAiSE) since 2004 [98]; and the international workshop on Quality of Information Systems (QoIS), held in conjunction with the Entity Relationship (ER) conference since 2005 [100]. There are also national conferences, held in France, Germany, and the US.

On the practical side, many data quality software tools are advertised and used in various data-driven applications, such as data warehousing, and to improve the quality of business processes. Frequently, their scope is limited and domain dependent, and it is not clear how to coordinate and finalize their use in data quality processes.

On the research side, the gap, still present between the need for techniques, methodologies, and tools, and the limited maturity of the area, has led so far to the presence of fragmented and sparse results in the literature, and the absence of a systematic view of the area.

Furthermore, in the area of data quality we highlight the existence of a dichotomy, typical of many other research areas that have a deep impact on real life, between practice-oriented approaches and formal research contributions. This book tries to address such a dichotomy, providing not only comparative overviews and explanatory frameworks of existing proposals, but also original solutions that combine the concreteness of practical approaches and the soundness of theoretical formalisms. By understanding the motivations and the different backgrounds of solutions, we have figured out the paradigms and forces contributing to the data quality environment.

Our main concern in this book is to provide a sound, integrated, and comprehensive picture of the state of the art and of future evolutions of data quality, in the database and information systems areas. This book includes an extensive description of techniques which constitute the core of data quality research, including record matching, data integration, error localization, and correction; such techniques are examined in a comprehensive and original methodological framework. Quality dimension definitions and adopted models are also deeply analyzed, and differences between the proposed solutions are highlighted and discussed. Furthermore, while systematically describing data quality as an autonomous research area, we highlight the paradigms and influences deriving from other areas, such as probability theory, statistical data analysis, data mining, knowledge representation, and machine learning. Our book also provides very practical solutions, such as methodologies, benchmarks for the most effective techniques, case studies, and examples.

The rigorous and formal foundation of our approach to data quality issues, presented with practical solutions, renders this book a necessary complement to books already published. Some books adopt a formal and research-oriented approach but are focused on specific topics or perspectives. Specifically, Dasu and Johnson [50] approach data quality problems from the perspective of data

mining and machine learning solutions. Wang et al. [206] provide a general perspective on data quality, by compiling a heterogeneous collection of contributions from different projects and research groups. Jarke et al. [104] describe solutions for data quality issues in the data warehouse environment. Wang et al. [203] is a survey of research contributions, including new methods for measuring data quality, for modeling quality improvement processes, and for organizational and educational issues related to information quality.

Some other books give much more room to practical aspects rather than to formal ones. In particular, leading books in the practitioners field are Redman' [167] and [169], and English' [68]. The two Redman' books provide an extensive set of data quality dimensions, and discuss a vast set of issues related to management methodologies for data quality improvement. English's book provides a detailed methodology for data quality measurement and improvement, discussing step-by-step issues related to data architectures, standards, process- and data-driven improvement methodologies, costs, benefits, and managerial strategies.

## Organization

The book is organized into nine chapters. Figure 0.1 lists the chapters and details interdependencies.

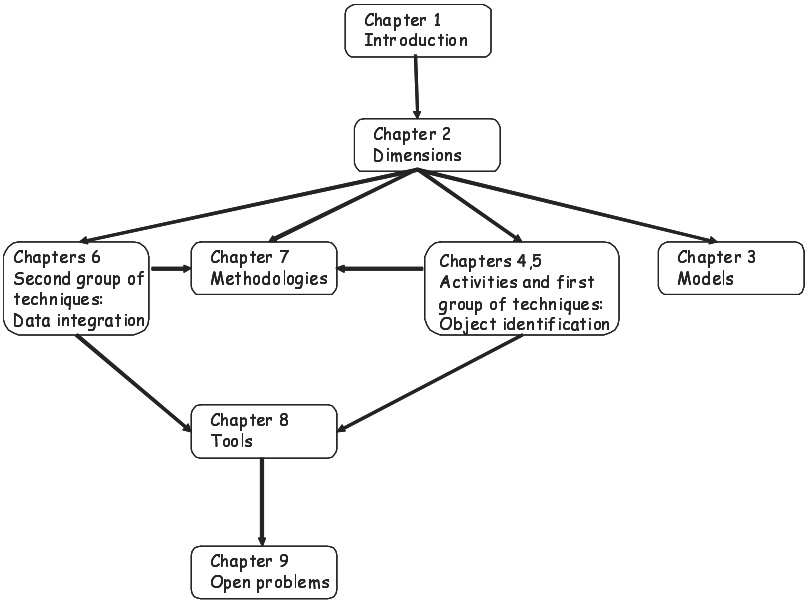



**Fig. 0.1.** Prerequisities among chapters

We initially provide basic concepts and establish coordinates to explore the area of data quality (Chapter 1). Then, we focus on dimensions that allow for the measurement of the quality of data values and data schemas (Chapter 2). These two chapters are preparatory to the rest of the book.

Models to express the quality of data in databases and information systems are investigated in Chapter 3. Chapter 4 describes the main activities for measuring and improving data quality. Some activities, such as error localization and correction, are introduced and fully described in Chapter 4; two specific chapters are dedicated to the most important activities and related research areas, namely object identification (Chapter 5) and data integration (Chapter 6), which are extensively investigated from the perspectives of relevant research paradigms and available techniques. Dimensions, models, activities, and techniques are the ingredients of any methodology for data quality measurement and improvement, and methodologies are the subject of Chapter 7. Specifically, in this chapter existing methodologies are examined and compared, and an original, comprehensive methodology is proposed, with an extensive case study. Tools, frameworks, and toolboxes proposed in the research literature for the effective use of techniques are described in Chapter 8. The book ends with Chapter 9, which puts all the ideas discussed in previous chapters in perspective and speculates on open problems and possible evolutions of the area.

## Intended Audience

The book is intended for those interested in a comprehensive introduction to the wide set of issues related to data quality. It has been written primarily for researchers in the fields of databases and information systems interested in investigating properties of data and information that have impact on the quality of processes and on real life. This book introduces the reader to autonomous research in the field of data quality, providing a wide spectrum of definitions, formalisms, and methods, with critical comparisons of the state of the art. For this reason, this book can help establish the most relevant research areas in data quality, consolidated issues and open problems.

A second category of potential readers are data and information system administrators and practitioners, who need a systematization of the field. This category also includes designers of complex cooperative systems and services, such as e-Business and e-Government systems, that exhibit relevant data quality problems.

Figures 0.2 and 0.3 suggest possible paths, which can be followed by the above audiences.

The *researcher path*, for researchers interested in the core research areas in data quality, skips chapters on methodologies (Chapter 7) and tools (Chapter 8). The *information system administrator path* skips models (Chapter 3), data integration issues (Chapter 6) and open problems (Chapter 9).

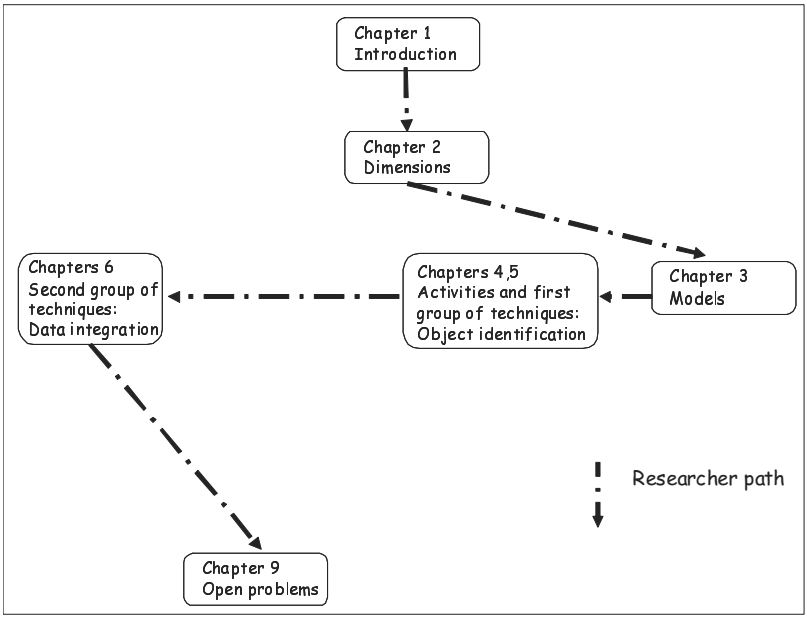



**Fig. 0.2.** Reading path for the researcher

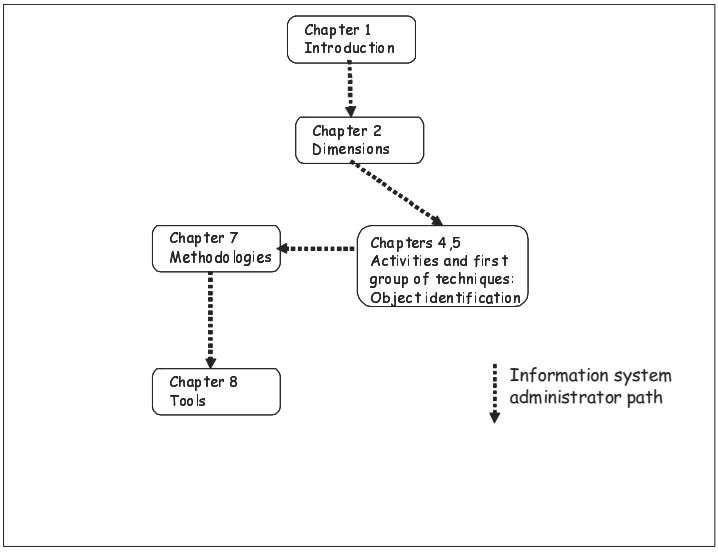



**Fig. 0.3.** Reading path for the information system administrator

## Guidelines for Teaching

To the best of our knowledge, data quality is not a usually considered topic in undergraduate and graduate courses. Several PhD courses include data quality issues, while the market for professional, often expensive courses is rapidly increasing. However, recent initiatives are in the direction of introducing data quality in undergraduate and graduate courses [1]. We have organized the book to be used in an advanced course on the quality of databases and information systems. The areas of databases and information systems are currently lacking consolidated textbooks on data quality; we have tried to cover this demand. Although this book cannot be defined a textbook, it can be adopted, with some effort, as basic material for a course in data quality. Due to the undeniable importance of these topics, what happened in the 1980's for other database areas, e.g., database design, could happen for data quality: the plethora of textbooks which favored the introduction of this area in university courses.

Data quality can be the topic of self-contained courses, or else of cycles of seminars in courses on databases and information systems management. Data integration courses would also benefit from data quality seminars. With regards to information systems management, data quality can be taught in connection with topics such as information management, information economics, business process reengineering, process and service quality, and cost and benefit analysis. Data quality techniques can be offered also in specific courses on data warehousing and data mining.

The material of this book is sufficiently self-contained for students who are able to attend a course in databases. As students' prerequisites, it is useful, but not mandatory, to have notions of mathematics and, to some extent, probability theory, statistics, machine learning, and knowledge representation.

The book provides enough material to cover all the necessary topics without the need for other textbooks. In the case of a PhD course, the references are a good starting point for assigning students in-depth analysis activities on specific issues.

In terms of exercises, a useful approach for students is to develop a complex data quality project that can be organized into two parts. The first part could be devoted to the assessment of the quality of two or more databases jointly used in several business processes of an organization. The second part could focus on the choice and application of methodologies and techniques described in Chapters 4, 5, 6, and 7 to improve data quality levels of the databases to a fixed target. This approach gives students a taste of the problems to face within a real-life environment.

[1] As an example, in 2005 the University of Arkansas at Little Rock promoted a Master of Science in Information Quality (MS IQ).

## Acknowledgements

First of all, we would like to thank several friends and researchers whose long-term frequentation and fruitful discussions on the topics covered in this book have positively influenced us; they are Daniele Barone, Laure Berti-Equille, Elisa Bertino, Paola Bertolazzi, Federico Cabitza, Cinzia Cappiello, Tiziana Catarci, Fabrizio De Amicis, Luca De Santis, Ahmed K. Elmagarmid, Markus Helfert, Domenico Lembo, Maurizio Lenzerini, Andrea Maurino, Massimo Mecella, Diego Milano, Paolo Missier, Felix Naumann, Davide Orlandi, Barbara Pernici, Louiqa Rashid, Ruggero Russo, Divesh Srivastava, Carlo Vaccari, and Richard Wang. Among students of our courses concerning data quality, we are particularly grateful to Matteo Strazzullo for his contributions on cost and benefit classifications.

Special mention should be given to the editors of this book series, Stefano Ceri and Mike Carey. We would also like to thank Ralf Gerstner of Springer-Verlag, who initially had the idea of considering data quality as a consolidated issue to be described in a book.

*Carlo Batini*
July 2006 *Monica Scannapieco*

# Contents

# 1

# Introduction to Data Quality

A Web search of the terms "data quality" through the search engine Google, returns about three millions of pages, an indicator that data quality issues are real and increasingly important (often, in the following, the term data quality will be shortened to the acronym DQ). The goal of this chapter is to introduce the relevant perspectives that make data quality an issue worth being investigated and understood. We first introduce the notion of data quality (Section 1.1), highlighting its relevance in real life and some of the main related initiatives in the public and private domains. Then, in Section 1.2, we show by means of several examples the multidimensional nature of data quality. Sections 1.3 and 1.4 analyze the different types of data, and the different types of information systems for which DQ can be investigated. In Section 1.5, we address the main research issues in DQ, application domains and related research areas. The research issues (Section 1.5.1) concern dimensions, models, techniques, methodologies, and tools; together, they provide the agenda for the rest of the book. Application domains are large sets, since data and information are fundamental ingredients of all the activities of people and organizations. We focus (Section 1.5.2) on three of the most relevant application domains, e-Government, Life Sciences, and the World Wide Web, highlighting the role that DQ plays in each of them. Research areas related to DQ will be examined in Section 1.5.3.

## 1.1 Why Data Quality is Relevant

The consequences of poor quality of data are often experienced in everyday life, but, often, without making the necessary connections to their causes. For example, the late or mistaken delivery of a letter is often blamed on a malfunctional postal service, although a closer look often reveals data-related causes, typically an error in the address, originating in the address database. Similarly, the duplicate delivery of automatically generated mail is often indicative of a database record duplication error.

Data quality has serious consequences, of far-reaching significance, for the efficiency and effectiveness of organizations and businesses. As already mentioned in the preface, the report on data quality of the Data Warehousing Institute (see [52]) estimates that data quality problems cost U.S. businesses more than 600 billion dollars a year. The findings of the report were based on interviews with industry experts, leading edge customers, and survey data from 647 respondents. In the following, we list further examples of the importance of data quality in organizational processes.

- *Customer matching.* Information systems of public and private organizations can be seen as the result of a set of scarcely controlled and independent activities producing several databases very often characterized by overlapping information. In private organizations, such as marketing firms or banks, it is not surprising to have several (sometimes dozens!) of customers registries, updated with different organizational procedures, resulting in inconsistent, duplicate information. As an example, it is very complex for banks to provide clients with a unique list of all their accounts and funds.
- *Corporate house-holding.* Many organizations establish separate relationships with single members of households, or, more generally, related groups of people; either way, they like, for marketing purposes, to reconstruct the household relationships in order to carry on more effective marketing strategies. This problem is even more complex than the previous one, since in that case the data to match concerned the same person, in this case it concerns groups of persons corresponding to the same household. For a detailed discussion on the relationship between corporate house holding information and various business application areas, see [200].
- *Organization fusion.* When different organizations or different units of an organization merge, it is necessary to integrate their legacy information systems. Such integration requires compatibility and interoperability at any layer of the information system, with the database level required to ensure both physical and semantic interoperability.

The examples above are indicative of the growing need to integrate information across completely different data sources, an activity in which poor quality hampers integration efforts. Awareness of the importance of improving the quality of data is increasing in many contexts. In the following, we summarize some of the major initiatives in both the private and public domains.

*Private Initiatives*

In the private sector, on the one hand, application providers and system integrators, and, on the other hand, direct users are experiencing the role of DQ in their own business processes.

With regard to application providers and systems integrators, IBM's recent (2005) acquisition of Ascential Software, a leading provider of data integration

tools, highlights the critical role data and information stewardship plays in the enterprise. The 2005 Ascential report [208] on data integration provides a survey that indicates data quality and security issues as the leading inhibitors (55 % of respondents in a multi-response survey) to successful data integration projects. The respondents also emphasize that data quality is more than just a technological issue. It requires senior management to treat data as a corporate asset and to realize that the value of this asset depends on its quality.

In the last few years, SAP [84] has set up a project for testing in the area of DQ and to build an internal methodology, with important savings (documented in [84]) in several internal business processes.

The awareness of the relevance of data quality issues has led Oracle (see [151]) to recently enhance its suite of products and services to support an architecture that optimizes data quality, providing a framework for the systematic analysis of data, with the goals of increasing the value of data, easing the burden of data migration, and decreasing the risks inherent in data integration.

With regard to users, Basel2 is an international initiative in the financial domain that requires financial services companies to have a risk sensitive framework for the assessment of regulatory capital. The planned implementation date for Basel2 is December 2006, with parallel operation from January 2006. The regulatory requirements of Basel2 are demanding improvements in data quality. For example, the Draft Supervisory Guidance on Internal Ratings-Based Systems for Corporate Credit states (see [19]): "institutions using the Internal Ratings-Based approach for regulatory capital purposes will need advanced data management practices to produce credible and reliable risk estimates"; and "data retained by the bank will be essential for regulatory risk-based capital calculations and public reporting. These uses underscore the need for a well defined data maintenance framework and strong controls over data integrity."

*Public Initiatives*

In the public sector a number of initiatives address data quality issues at international, European, and national levels. We focus in the rest of the section on two of the main initiatives, the Data Quality Act in the US and the European directive on reuse of public data.

In 2001 the President of the US signed into law important new Data Quality legislation, concerning "Guidelines for Ensuring and Maximizing the Quality, Objectivity, Utility, and Integrity of Information Disseminated by Federal Agencies," in short the Data Quality Act. The Office of Management and Budget (OMB) issued guidelines referred for policies and procedures on data quality issues (see [149]). Obligations mentioned in the guidelines concern agencies, which are to report periodically to the OMB regarding the number and nature of data quality complaints received, and how such complaints were handled. OMB must also include a mechanism through which

the public can petition agencies to correct information that does not meet the OMB standard. In the OMB guidelines data quality is defined as an encompassing term comprising utility, objectivity, and integrity. Objectivity is a measure to determine whether the disseminated information is accurate, reliable, and unbiased, and whether that information is presented in an accurate, clear, complete, and unbiased manner. Utility refers to the usefulness of the information for its anticipated purpose, by its intended audience. OMB is committed to disseminating reliable and useful information. Integrity refers to the security of information, namely protection of the information from unauthorized, unanticipated, or unintentional modification, to prevent it from being compromised by corruption or falsification. Specific risk-based, cost-effective policies are defined for assuring integrity.

The European directive 2003/98/CE on the reuse of public data (see [71]) highlights the importance of reusing the vast data assets owned by public agencies. The public sector collects, produces, and disseminates a wide range of information in many areas of activity, such as social, economic, geographical, metereological, business, and educational information. Making public all generally available documents held by the public sector, concerning not only the political process but also the legal and administrative processes, is considered a fundamental instrument for extending the right to information, which is a basic principle of democracy. Aspects of data quality addressed by such a directive are the accessibility of public data and availability in a format which is not dependent on the use of specific software. At the same time, a related and necessary step for public data reuse is to guarantee its quality in terms of accuracy and currency, through data cleaning campaigns. This makes it attractive to new potential users and customers.

## 1.2 Introduction to the Concept of Data Quality

From a research perspective, data quality has been addressed in different areas, including statistics, management, and computer science. Statisticians were the first to investigate some of the problems related to data quality, by proposing a mathematical theory for considering duplicates in statistical data sets, in the late 1960's. They were followed by researchers in management, who at the beginning of the 1980's focused on how to control data manufacturing systems in order to detect and eliminate data quality problems. Only at the beginning of the 1990's computer scientists begin considering the problem of defining, measuring, and improving the quality of electronic data stored in databases, data warehouses, and legacy systems.

When people think about data quality, they often reduce data quality just to accuracy. For example, let us consider the surname "Batini"; when this is spelled during a telephone call, several misspellings are reported by the other side, such as "Vatini," "Battini," "Barini," "Basini," all inaccurate versions of the original last name. Indeed, data are normally considered to be of poor

quality if typos are present or wrong values are associated with a concept instance, such as an erroneous birth date or age associated with a person. However, data quality is more than simply data accuracy. Other significant dimensions such as completeness, consistency, and currency are necessary in order to fully characterize the quality of data. In Figure 1.1 we provide some examples of these dimensions, which are described in more detail among others in Chapter 2. The relation in the figure describes movies, with title, director, year of production, number of remakes, and year of the last remake.

| Id | Title | Director | Year | #Remakes | LastRemakeYear |
|---|---|---|---|---|---|
| 1 | Casablanca | Weir | 1942 | 3 | 1940 |
| 2 | Dead poets society | Curtiz | 1989 | 0 | NULL |
| 3 | Rman Holiday | Wylder | 1953 | 0 | NULL |
| 4 | Sabrina | null | 1964 | 0 | 1985 |

**Fig. 1.1.** A relation `Movies` with data quality problems

In the figure, the cells with data quality problems are shaded. At first, only the cell corresponding to the title of movie 3 seems to be affected by a data quality problem. In fact, there is a misspelling in the title, where `Rman` stands for `Roman`, thus causing an accuracy problem. Nevertheless, another accuracy problem is related to the exchange of the director between movies 1 and 2; Weir is actually the director of movie 2 and Curtiz the director of movie 1. Other data quality problems are a missing value for the director of movie 4, causing a completeness problem, and a 0 value for the number of remakes of movie 4, causing a currency problem because a remake of the movie has actually been proposed. Finally, there are two consistency problems: first, for movie 1, the value of `LastRemakeYear` cannot be lower than `Year`; second, for movie 4 the value of `LastRemakeYear` cannot be different from null, because the value of `#Remakes` is 0.

The above examples of dimensions concern the *quality of data* represented in the relation. Besides data, a large part of the design methodologies for the relational model addresses properties that concern the *quality of the schema*; for example, several normal forms have been proposed with the aim of capturing the concept of good relational schema, free of anomalies and redundancies. For instance, the relational schema of Figure 1.1 is in the Boyce Codd normal form, since all attributes that do not belong to a superkey are functionally dependent on the superkeys (`Id` and `Title`). Other data quality and schema quality dimensions will be discussed in Chapter 2. The above examples and considerations show that:

- Data quality is a multifaceted concept, as in whose definition different dimensions concur.
- The quality dimensions, e.g., accuracy, can be easily detected in some cases (e.g., misspellings) but are more difficult to detect in other cases (e.g., where admissible but not correct values are provided).
- A simple example of a completeness error has been shown, but as with accuracy, completeness can also be very difficult to evaluate (e.g., if a tuple representing a movie is entirely missing from the relation `Movie`).
- Consistency detection does not always localize the errors (e.g., for movie 1, the value or the `LastRemakeYear` attribute is wrong).

The above example concerned a relational table of a single database. Problems change significantly when other *types of data* are involved, and more complex *types of information systems* than a single database are considered. We now address these two aspects.

## 1.3 Data Quality and Types of Data

Data represent real world objects, in a format that can be stored, retrieved, and elaborated by a software procedure, and communicated through a network. The process of representing the real world by means of data can be applied to a large number of phenomena, such as measurements, events, characteristics of people, the environment, sounds, and smells. Data are extremely versatile in such representation. Besides data, other types of *information* are used in real-life and business processes, such as paper-based information, and information conveyed by the voice. We will not deal with all these types of information, and we concentrate on data.

Since researchers in the area of data quality must deal with a wide spectrum of possible data representations, they have proposed several classifications for data. First, several authors distinguish, implicitly or explicitly, three types of data:

1. *Structured*, when each data element has an associated fixed structure. Relational tables are the most popular type of structured data.
2. *Semistructured*, when data has a structure which has some degree of flexibility. Semistructured data are also "schemaless" or "self-describing" (see [1], [35], and [40]). XML is the markup language commonly used to represent semistructured data. Some common characteristics are (i) data can contain fields not known at design time; for instance, an XML file does not have an associated XML schema file; (ii) the same kind of data may be represented in multiple ways; for example, a date might be represented by one field or by multiple fields, even within a single set of data; and (iii) among fields known at design time, many fields will not have values.
3. *Unstructured*, when data are expressed in natural language and no specific structure or domain types are defined.

It is intuitive that dimensions and techniques for data quality have to be adapted for the three types of data described above, and are progressively more complex to conceive and use from structured to unstructured data.

A second point of view sees data as a product. This point of view is adopted, for example, in the IP-MAP model (see [177]), an extension of the Information Manufacturing Product model [201], which will be discussed in detail in Section 3.4; the IP-MAP model identifies a parallelism between the quality of data, and the quality of products as managed by manufacturing companies. In this model, three different types of data are distinguished:

- *raw data items* are considered smaller data units. They are used to construct information and component data items that are semi-processed information;
- while the raw data items may be stored for long periods of time, the *component data items* are stored temporarily until the final product is manufactured. The component items are regenerated each time an information product is needed. The same set of raw data and component data items may be used (sometimes simultaneously) in the manufacturing of several different products;
- *information products*, which are the result of a manufacturing activity performed on data.

Looking at data as a product, as discussed in Chapters 3 and 7, methodologies and procedures used over a long period, with suitable changes having been made to them, can be applied to data for quality assurance in manufacturing processes.

The third classification, proposed in [133], addresses a typical distinction made in information systems between elementary data and aggregated data. *Elementary* data are managed in organizations by operational processes, and represent atomic phenomena of the real world (e.g., social security number, age, sex). *Aggregated* data are obtained from a collection of elementary data by applying some aggregation function to them (e.g., the average income of tax payers in a given city). This classification is useful to distinguish different levels of severity in measuring and achieving the quality of data. As an example, the accuracy of an attribute `Sex` changes dramatically if we input `M` (male) instead of `F` (female); if the age of a single person is wrongly recorded as 25 instead of 35, the accuracy of the average age of a population of millions of inhabitants is minimally affected.

Dasu and Johnson in [50] investigate new types of data that emerge from the diffusion of networks and Internet, and observe that the definition of data itself has changed dramatically to include "any kind of information that is analyzed systematically." They distinguish several new types of data, among them are relevant in this book:

- *federated data*, which come from different heterogeneous sources, and, consequently, require disparate data sources to be combined with approximate matches;

- *web data*, that are "scraped" from the Web and, although characterized by unconventional formats and low control on data, more often constitute the primary source of information for several activities.

Previous classifications were not interested in the time dimension of data, investigated in [30]. According to its change frequency, we can classify source data into three categories:

- *stable* data is data that is unlikely to change. Examples are scientific publications; although new publications can be added to the source, older publications remain unchanged;
- *long-term-changing data* is data that has very low change frequency. Examples are addresses, currencies, and hotel price lists. The concept of low frequency is domain dependent; in an e-trade application, if the value of a stock quote is tracked once an hour, it is considered to be a low frequency change, while a shop that changes its goods weekly has a high-frequency change for clients;
- *frequently-changing data* is data that has intensive change, such as real-time traffic information, temperature sensor measures, and sales quantities. The changes can occur with a defined frequency or they can be random.

For this classification, the procedures for establishing the time dimension qualities of the three types of data, i.e., stable, long-term-changing, and frequently-changing data, are increasingly more complex.

Among the different types of data resulting from the above classification, we are mainly interested in focusing our attention on *structured* and *semistructured elementary data*, and on *information products*. Such types of data have been deeply investigated in the literature, and, to a certain extent, consolidated techniques and methodologies have been concieved. This does not mean that we will exclude other types of data from our analysis: dimensions for time-dependent data will be introduced and discussed in Chapter 2, and web data will be considered in Chapter 9, dedicated to open problems.

As a terminological note, when we give generic examples of structured data, we use the term *tuple* to indicate a set of *fields* or *cell values*, corresponding usually to different *definition domains* or *domains*, describing properties or *attributes* of a specific real world object; we use interchangeably the terms *relational table* or *table* or *relation* to indicate a set of tuples. As a consequence, *tuple* can be used in place of *record* and *table*/*relation* can be used in place of *structured file*. When we refer to generic data, we use the term *record* to indicate a set of fields, and we use interchangeably the terms *file* or *data set* to indicate a set of tuples.

## 1.4 Data Quality and Types of Information Systems

Data are collected, stored, elaborated, retrieved, and exchanged in *information systems* used in organizations to provide services to business processes. Different criteria can be adopted for classifying the different types of information systems, and their corresponding architectures; they are usually related to the overall organizational model adopted by the organization or the set of the organizations that make use of the information system. In order to clarify the impact of data quality on the different *types of information systems*, we adapt the classification criteria proposed in [153] for distributed databases. Three different criteria are proposed: distribution, heterogeneity, and autonomy.

*Distribution* deals with the possibility of distributing the data and the applications over a network of computers. For simplicity, we adopt a <`yes`, `no`> domain for distribution. *Heterogeneity* considers all types of semantic and technological diversities among systems used in modeling and physically representing data, such as database management systems, programming languages, operating systems, middleware, markup languages. For heterogeneity we also adopt a simple <`yes`,`no`> domain. *Autonomy* has to do with the degree of hierarchy and rules of coordination, establishing rights and duties, defined in the organization using the information system. The two extremes are: (i) a fully hierarchical system, where only one subject decides for all, and no autonomy at all exists; and (ii) a total anarchy, where no rule exists, and each component organization is totally free in its design and management decisions. In this case we adopt a three-value <`no, semi, totally`> domain.

The three classifications are represented together in the classification space of Figure 1.2. Among all possible combinations, five main types of information systems are highlighted in the figure: Monolithic, Distributed, Data Warehouses, Cooperative, and Peer-to-Peer.

- In a *monolithic information system* presentation, application logic, and data management are merged into a single computational node. Many monolithic information systems are still in use. While being extremely rigid, they provide advantages to organizations, such as reduced costs due to the homogeneity of solutions and centralization of management. In monolithic systems, data flows have a common format, and data quality control is facilitated by the homogeneity and centralization of procedures and management rules.
- A *data warehouse* (DW) is a centralized set of data collected from different sources, designed to support management decision making. The most critical problem in DW design concerns the cleaning and integration of the different data sources that are loaded into the DW, in that much of the implementation budget is spent on data cleaning activities.
- A *distributed information system* relaxes the rigid centralization of monolithic systems, in that it allows the distribution of resources and applications across a network of geographically distributed systems. The network

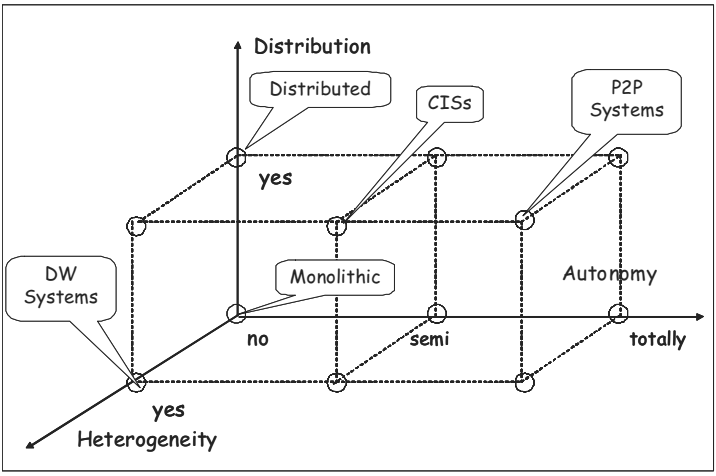



**Fig. 1.2.** Types of information systems

can be organized in terms of several tiers, each made of one or more computational nodes. Presentation, application logic, and data management are distributed across tiers. Usually, the different tiers and nodes have a limited degree of autonomy, data design is usually performed centrally, but to a certain extent some degree of heterogeneity can occur, due to the impossibility of establishing unified procedures. Problems of data management are more complex than in monolithic systems, due to the reduced level of centralization. Heterogeneities and autonomy usually increase with the number of tiers and nodes.

- A *cooperative information system* (CIS) can be defined as a large-scale information system that interconnects various systems of different and autonomous organizations, while sharing common objectives. According to [58], the manifesto of cooperative information systems, "an information system is cooperative if it shares goals with other agents in its environment, such as other information systems, human agents, and the organization itself, and contributes positively toward the fulfillment of these common goals." The relationship between cooperative information systems and DQ is double-faced: on the one hand it is possible to profit the cooperation between agents in order to choose the best quality sources, and thus improve the quality of circulating data. On the other hand, data flows are less controlled than in monolithic systems, and the quality of data, when not controlled, may rapidly decrease in time. Integration of data sources is also a relevant issue in CISs, especially when partners decide to substitute a group of databases, that have been independently developed, with an integrated in-house database. In *virtual data integration* a unique virtual integrated schema is built to provide unified access. This case is affected by

data quality problems, because inconsistencies in data stored at different sites make it difficult to provide integrated information.

- In a *peer-to-peer information system* (usually abbreviated P2P), the traditional distinction between clients and servers typical of distributed systems is disappearing. A P2P system can be characterized by a number of properties: peers are higly autonomous and higly heterogeneous, they have no obligation for the quality of their services and data, no central coordination and no central database exist, no peer has a global view of the system, global behavior emerges from local interactions. It is clear that P2P systems are extremely critical from the point of view of data quality, since no obligation exists for agents participating in the system. It is also costly for a single agent to evaluate the reputation of other partners.

In the rest of the book, we will examine DQ issues mainly conceived for monolithic, distributed, data warehouses, and cooperative information systems, while issues for P2P systems will be discussed in Chapter 9 on open problems.

## 1.5 Main Research Issues and Application Domains in Data Quality

Due to the relevance of data quality, its nature, and the variety of data types and information systems, achieving data quality is a complex, multidisciplinary area of investigation. It involves several research topics and real-life application areas. Figure 1.3 shows the main ones.

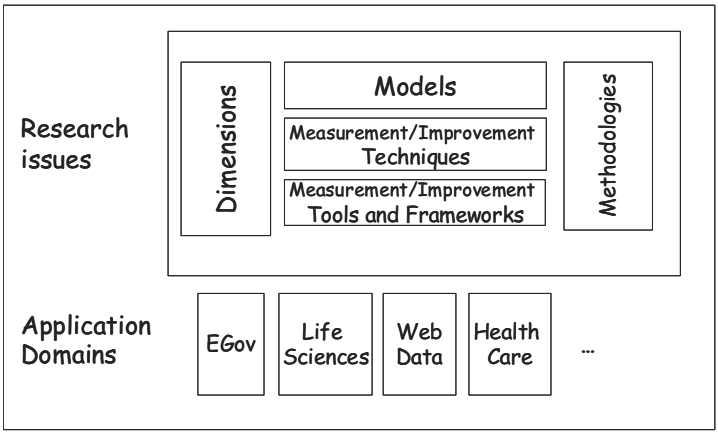



**Fig. 1.3.** Main issues in data quality

Research issues concern models, techniques, and tools, and two "vertical" areas, that cross the first three, i.e. dimensions and methodologies. We will

discuss them in Section 1.5.1. Three of the application domains mentioned in Figure 1.3, namely e-Government, Life Sciences, and the World Wide Web, in which DQ is particularly relevant, are discussed in Section 1.5.2.

Research issues in DQ originate from research paradigms initially developed in other areas of research. The relationship between data quality and these related research areas will be discussed in Section 1.5.3.

### 1.5.1 Research Issues in Data Quality

Choosing *dimensions* to measure the level of quality of data is the starting point of any DQ-related activity. Though measuring the quality of ICT technologies, artifacts, processes, and services is not a new issue in research, for many years several standardization institutions have been operating (e.g. ISO, see [97]) in order to establish mature concepts in the areas of quality characteristics, measurable indicators, and reliable measurement procedures. Dimensions are discussed in Chapter 2. Dimensions are applied with different roles in models, techniques, tools, and frameworks.

*Models* are used in databases to represent data and data schemas. They are also used in information systems to represent business processes of the organization; these models have to be enriched in order to represent dimensions and other issues related to DQ. Models are investigated in Chapter 3.

*Techniques* correspond to algorithms, heuristics, knowledge-based procedures, and learning processes that provide a solution to a specific DQ problem or, as we say, to a *data quality activity*, as defined in Chapter 4. Examples of DQ activities are identifying if two records of different databases represent the same object of the real world or not; or finding the most reliable source for some specific data. DQ activities are defined in Chapter 4 and tecniques are discussed in Chapters 4, 5, and 6.

*Methodologies* provide guidelines to choose, starting from available techniques and tools, the most effective DQ measurement and improvement process (and hopefully, most economical for comparable results) within a specific information system. Methodologies are investigated in Chapter 7.

Methodologies and techniques, in order to be effective, need the support of *tools*, i.e., automatized procedures, provided with an interface, that relieve the user of the manual execution of some techniques. When a set of coordinated tools is integrated to provide a set of DQ services, we will use the term *framework*. Tools and frameworks are discussed in Chapter 8.

### 1.5.2 Application Domains in Data Quality

In this section, we analyze three distinct application domains of DQ. Their importance has been growing over the last few years, because of their relevance in daily lives of people and organizations: e-Government, Life Sciences, the World Wide Web.

### e-Government

The main goal of all e-Government projects is the improvement of the relationship between the government, agencies, and citizens, as well as between agencies and businesses, through the use of information and communication technologies. This ambitious goal is articulated in different objectives:

1. the complete automation of those government administrative processes that deliver services to citizens and businesses, and that involve the exchange of data between government agencies;
2. the creation of an architecture that, by connecting the different agencies, enables them to fulfill their administrative processes without any additional burden to the users that benefit from them; and
3. the creation of portals that simplify access to services by authorized users.

e-Government projects must face the problem that similar information about one citizen or business is likely to appear in multiple databases. Each database is autonomously managed by the different agencies that historically has never been able to share data about citizens and businesses.

The problem is worsened by the many errors usually present in the databases, for many reasons. First, due to the nature of the administrative flows, several citizens' data (e.g. addresses) are not updated for long periods of time. This happens because it is often impractical to obtain updates from subjects that maintain the official residence data. Also, errors may occur when personal data on individuals is stored. Some of these errors are not corrected and a potentially large fraction of them is not detected. Furthermore, data provided by distinct sources differ in format, following local conventions, that can change in time and result in multiple versions. Finally, many of the records currently in the database were entered over years using legacy processes that included one or more manual data entry steps.

A direct consequence of this combination of redundancy and errors in data is frequent mismatches between different records that refer to the same citizen or business. One major outcome of having multiple disconnected views for the same information is that citizens and businesses experience consistent service degradation during their interaction with the agencies. Furthermore, misalignment brings about additional costs. First, agencies must make an investment to reconcile records using clerical review, e.g., to manually trace citizens and businesses that cannot be correctly and unequivocally identified. Secondly, because most investigation techniques, e.g., tax fraud prevention techniques, rely on cross-referencing records of different agencies, misalignment results in undetected tax fraud and reduced revenues.

### Life Sciences

Life sciences data and specifically biological data are characterized by a diversity of data types, very large volumes, and highly variable quality. Data

are available through vastly disparate sources and disconnected repositories. Their quality is difficult to assess and often unacceptable for the required usage. Biologists typically search several sources, for good quality data, for instance, in order to perform reliable in-silico experiments. However, the effort to actually assess the quality level of such data is entirely in the hands of the biologists; they have to manually analyze disparate sources, trying to integrate and reconcile heterogeneous and contradictory data in order to identify the best information. Let us consider, as an example, a gene analysis scenario. Figure 1.4 shows an example of a simple data analysis pipeline. As the result of a micro-array experiment, a biologist wants to analyze a set of genes, with the objective of understanding their functions.

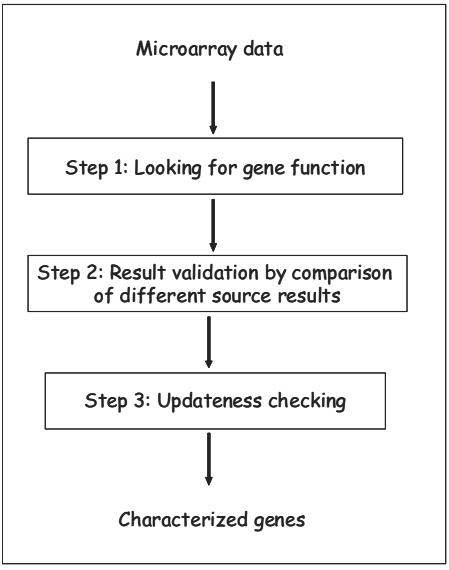



**Fig. 1.4.** Example of biological data analysis process

In Step 1, the biologist performs a Web search on a site that is known to contain gene data for the particular organism under consideration. Once the data is obtained, the biologist must assess its reliability. Therefore, in Step 2 the biologist performs a new web search in order to check if other sites provide the same gene information. It may happen that different sites provide conflicting results. Then (Step 3) the biologist also has to check that the provided results are up-to-date, i.e., if a gene is unknown in the queried sites, or no recent publication on that gene is available, e.g. through Pubmed (see [192]). The described scenario has many weaknesses:

1. the biologist must perform a time-consuming manual search for all the sources that may provide the function of the interested gene. This process

is also dependent on the user having personal knowledge about which sites must be queried;
2. the biologist has no way of assessing the trustworthiness of a result;
3. in Step 2, the biologist has no way of evaluating the quality of the results provided by different sites.
4. in Step 3, a new web search must be performed which again can be very time consuming.

In order to overcome such weaknesses, life sciences and biology need robust data quality techniques.

### World Wide Web

Web information systems are characterized by the presentation of a large amount of data to a wide audience, the quality of which can be very heterogeneous. There are several reasons for this variety. First, every organization and individual can create a Web site and load every kind of information without any control on its quality, and sometimes with a malicious intent. A second reason lies in the conflict between two needs. On the one hand information systems on the web need to publish information in the shortest possible time after it is available from information sources. On the other hand, information has to be checked with regard to its accuracy, currency, and trustworthiness of its sources. These two requirements are in many aspects contradictory: accurate design of data structures, and, in the case of Web sites, of good navigational paths between pages, and certification of data to verify its correctness are costly and lengthy activities. However, the publication of data on Web sites is subject to time constraints.

Web information systems present two further aspects in connection to data quality that differentiate them from traditional information sources: first, a Web site is a continuously evolving source of information, and it is not linked to a fixed release time of information; second, in the process of changing information, additional information can be produced in different phases, and corrections to previously published information are possible, creating, in such a way, further needs for quality checks. Such features lead to a different type of information than with traditional media.

As a final argument, in Web information systems it is practically impossible to individuate a subject, usually called *data owner*, responsible for a certain data category. In fact, data are typically replicated among the different participating organizations, and one does not know how to state that an organization or subject has the primary responsibility for some specific data.

All previously discussed aspects make it difficult to certify the quality of data sources, and, for a user, to assess the reputation of other users and sources.

### 1.5.3 Research Areas Related to Data Quality

Data quality is fairly a new research area. Several other areas (see Figure 1.5) in computer science and other sciences have in the past treated related and overlapping problems; at the same time, such areas have developed in the last decades (in the case of statistics, in the last centuries) paradigms, models, and methodologies that have proved to be of major importance in grounding the data quality research area. We now discuss such research areas.

1. *Statistics* includes a set of methods that are used to collect, analyze, present, and interpret data. Statistics has developed in the last two centuries a wide spectrum of methods and models that allow one to express predictions and formulate decisions in all contexts where uncertain and imprecise information is available for the domain of interest. As discussed in [121], statistics and statistical methodology as the basis of data analysis are concerned with two basic types of problems: (i) summarizing, describing, and exploring data, (ii) using sampled data to infer the nature of the process that produced the data. Since low quality data are an inaccurate representation of the reality, a variety of statistical methods have been developed for measuring and improving the quality of data. We will discuss some statistical methods in Chapters 4 and 5.
2. *Knowledge representation* (see [144] and [54] for insightful introductions to the area) is the study of how knowledge about an application domain can be represented, and what kinds of reasoning can be done with that knowledge (this is called *knowledge reasoning*). Knowledge about an application domain may be represented procedurally in form of program code, or implicitly as patterns of activation in a neural network. Alternatively, the area of knowledge representation assumes an explicit and declarative representation, in terms of a *knowledge base*, consisting of logical formulas or rules expressed in a representation language. Providing a rich representation of the application domain, and being able to reason about it, is becoming an important leverage in many techniques for improving data quality; we will see some of these techniques in Chapters 5 and 8.
3. *Data mining* (see [92]) is an analytic process designed to explore usually large sets of data in search of consistent patterns and/or systematic relationships between attributes/variables. *Exploratory data mining* is defined in [50] as the preliminary process of discovering structure in a set of data using statistical summaries, visualization, and other means. In this context, achieving good data quality is an intrinsic objective of any data mining activity (see [46]), since otherwise the process of discovering patterns, relationships and structures is seriously deteriorated. From another perspective, data mining techniques may be used in a wide spectrum of activities for improving the quality of data; we will examine some of them in Chapter 4.

4. *Management information systems* (see [53]) are defined as systems that provide the information necessary to manage an organization effectively. Since data and knowledge are becoming relevant resources both in operational and decision business processes, and poor quality data result in poor quality processes, it is becoming increasingly important to supply management information systems with functionalities and services that allow one to control and improve the quality of the data resource.
5. *Data integration* (see [116]) has the goal of building and presenting a unified view of data owned by heterogeneous data sources in distributed, cooperative, and peer-to-peer information systems. Data integration will be considered in Chapter 4 as one of basic activities whose purpose is improving data quality, and will be discussed in detail in Chapter 6. While being an autonomous and well-grounded research area, data integration will be considered in this book as strictly related to data quality, regarding two main issues, providing query results on the basis of a quality characterization of data at sources, and identifying and solving conflicts on values referring to the same real-world objects.

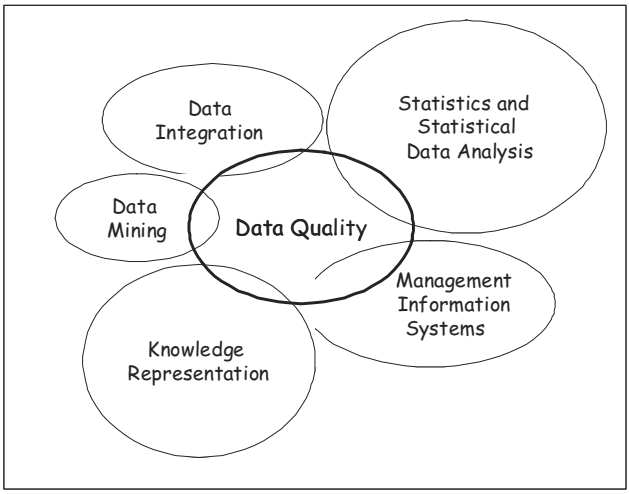



**Fig. 1.5.** Research areas related to data quality

## 1.6 Summary

In this chapter we have perceived that data quality is a multidisciplinary area. This is not surprising, since data, in a variety of formats and with a variety of media, are used in every real-life or business activity, and deeply influence

the quality of processes that use data. Many private and public organizations have perceived the impact of data quality on their assets and missions, and have consequently launched initiatives of large impact. At the same time, while in monolithic information systems data are processed within controlled activities, with the advent of networks and the Internet, data are created and exchanged with much more "turbulent" processes, and need more sophisticated management.

The issues discussed in this chapter introduce to the structure of the rest of the book: dimensions, models, techniques, methodologies, tools, and frameworks will be the main topics addressed. While data quality is a relatively new research area, other areas, such as statistical data analysis, have addressed in the past some aspects of the problems related to data quality; with statistical data analysis, also knowledge representation, data mining, management information systems, and data integration share some of the problems and issues characteristic of data quality, and, at the same time, provide paradigms and techniques that can be effectively used in data quality measurement and improvement activities.

# 2

# Data Quality Dimensions

In Chapter 1 we provided an intuitive concept of data quality and we informally introduced several data quality dimensions, such as accuracy, completeness, currency, and consistency.

This chapter investigates data quality in more depth, and presents multiple associated *dimensions*. Each dimension captures a specific aspect included under the general umbrella of data quality. Both data and schema dimensions are important. Data of low quality deeply influences the quality of business processes, while a schema of low quality, e.g., an unnormalized schema in the relational model, results in potential redundancies and anomalies during the lifecycle of data usage. Data dimensions can be considered more relevant in real-life applications and processes than schema dimensions.

More specifically, quality dimensions can refer either to the *extension* of data, i.e., to data values, or to their *intension*, i.e., to their schema. Both data dimensions and schema dimensions are usually defined in a qualitative way, referring to general properties of data and schemas, and the related definitions do not provide any facility for assigning values to dimensions themselves. Specifically, definitions do not provide quantitative measures, and one or more *metrics* are to be associated with dimensions as separate, distinct properties. For each metric, one or more *measurement methods* are to be provided regarding (see [169]) (i) where the measurement is taken, (ii) what data are included, (iii) the measurement device, and (iv) the scale on which results are reported. According to the literature, at times we will distinguish between dimensions and metrics, while other times we will directly provide metrics.

The quality of conceptual and logical schemas is very important in database design and usage. Conceptual schemas are typically produced within the first phase of the development of an information system. Erroneous conceptual schema design strongly impacts the system development and must be detected as soon as possible. Logical schemas are at the base of the implementation of any database application. Methods and techniques for assessing, evaluating, and improving conceptual schemas and logical schemas in different application domains is still a fertile research area.

Despite such recognized importance, the prevalent attention to the definitions of data quality dimensions has been devoted to data values, which, more extensively than schemas, are used in business and administrative processes. As a consequence, in this chapter we deal especially with data dimensions, but we also discuss some of the most relevant schema dimensions.

In the following sections we describe in detail data dimensions in order to understand the different possible meanings and metrics. Some definitions of data dimensions are independent of the data model used to represent the data. Specifically, the definitions provided for accuracy and time-related dimensions are model independent. Where some specific features of dimensions will require referring to the data model, it will be explicitly highlighted. Most examples refer to the relational model, and thus the relational terminology introduced in Chapter 1 is adopted. More specifically, we provide detailed descriptions of accuracy (Section 2.1), completeness (Section 2.2), currency and other time dimensions (Section 2.3), and finally consistency (Section 2.4). Section 2.5 discusses other dimensions which are mainly related to the evolution of information systems toward networked, web-based information systems. Some proposals of comprehensive classifications of dimensions are first illustrated and then compared in Section 2.6. Section 2.7 deals with schema dimensions, briefly describing correctness, minimality, completeness, and pertinence, and, in more detail, readability and normalization.

## 2.1 Accuracy

*Accuracy* is defined as the closeness between a value $\mathtt{v}$ and a value $\mathtt{v}'$, considered as the correct representation of the real-life phenomenon that $\mathtt{v}$ aims to represent. As an example if the name of a person is `John`, the value $\mathtt{v}' = \mathtt{John}$ is correct, while the value $\mathtt{v} = \mathtt{Jhn}$ is incorrect. Two kinds of accuracy can be identified, namely a syntactic accuracy and a semantic accuracy.

*Syntactic accuracy* is the closeness of a value $\mathtt{v}$ to the elements of the corresponding definition domain D. In syntactic accuracy we are not interested in comparing $\mathtt{v}$ with the true value $\mathtt{v}'$; rather, we are interested in checking whether $\mathtt{v}$ is any one of the values in D, whatever it is. So, if $\mathtt{v} = \mathtt{Jack}$, even if $\mathtt{v}' = \mathtt{John}$, $\mathtt{v}$ is considered syntactically correct, as `Jack` is an admissible value in the domain of persons' names. Syntactic accuracy is measured by means of functions, called *comparison functions*, that evaluate the distance between $\mathtt{v}$ and the values in D. Edit distance is a simple example of a comparison function, taking into account the minimum number of character insertions, deletions, and replacements to convert a string $\mathtt{s}$ to a string $\mathtt{s}'$. More complex comparison functions exist, for instance taking into account similar sounds or character transpositions. In Chapter 5, a detailed description of the main comparison functions will be provided.

Let us consider the relation `Movies` introduced in Chapter 1, shown in Figure 2.1.

| Id | Title | Director | Year | #Remakes | LastRemakeYear |
|---|---|---|---|---|---|
| 1 | Casablanca | Weir | 1942 | 3 | 1940 |
| 2 | Dead Poets Society | Curtiz | 1989 | 0 | NULL |
| 3 | Rman Holiday | Wylder | 1953 | 0 | NULL |
| 4 | Sabrina | NULL | 1964 | 0 | 1985 |

**Fig. 2.1.** A relation `Movies`

The value `Rman Holiday` in movie 3 for `Title` is syntactically inaccurate, since it does not correspond to any title of a movie. `Roman Holiday` is the closest movie name to `Rman Holiday`; indeed, the edit distance between `Rman Holiday` and `Roman Holiday` is equal to 1 and simply corresponds to the insertion of the char `o` in the string `Rman Holidays`. Since 1 is the edit distance, the measure of syntactic accuracy is 1. More precisely, given a comparison function `C`, we may define a measure of syntactic accuracy of a value `v` with respect to a definition domain `D`, as the minimum value of `C`, when comparing `v` with all the values in `D`. Such a measure will be in the domain [0, ..., n], where `n` is the maximum possible value that the comparison function may have.

*Semantic accuracy* is the closeness of the value `v` to the true value `v′`. Let us consider again the relation `Movies` of Figure 2.1. The exchange of directors' names in tuples 1 and 2 is an example of a semantic accuracy error: indeed, for movie 1, a director named `Curtiz` would be admissible, and thus it is syntactically correct. Nevertheless, `Curtiz` is not the director of `Casablanca`; therefore a semantic accuracy error occurs.

The above examples clearly show the difference between syntactic and semantic accuracy. Note that, while it is reasonable to measure syntactic accuracy using a distance function, semantic accuracy is measured better with a <`yes, no`> or a <`correct, not correct`> domain. Consequently, semantic accuracy coincides with the concept of *correctness*. In contrast with what happens for syntactic accuracy, in order to measure the semantic accuracy of a value `v`, the corresponding true value has to be known, or, else, it should be possible, considering additional knowledge, to deduce whether that the value `v` is or is not the true value.

From the above arguments, it is clear that semantic accuracy is typically more complex to calculate than syntactic accuracy. When it is known a priori that the rate of errors is low, and the errors result typically from typos, then syntactic accuracy tends to coincide with semantic accuracy, since typos produce values close to the true ones. As a result, semantic accuracy may be achieved by replacing an inaccurate value with the closest value in the definition domain, under the assumption that it is the true one.

In a more general context, a technique for checking semantic accuracy consists of looking for the same data in different data sources and finding the correct data by comparisons. This latter approach also requires the solution of the *object identification problem*, i.e., the problem of understanding whether two tuples refer to the same real-world entity or not; this problem will be discussed extensively in Chapter 5. The main issues to be addressed for solving the object identification problem are

- *Identification*: tuples in one or several sources may not have unique identifiers, and thus they need to be put in correspondence by means of appropriate *matching keys*.
- *Decision strategy*: once tuples are linked on the basis of a matching key, a decision must be made to state whether it corresponds to the same entity or not.

The accuracy above discussed is referred to a single value of a relation attribute. In practical cases, coarser accuracy definitions and metrics may be applied. As an example, it is possible to calculate the accuracy of an attribute called *attribute* (or *column*) *accuracy*, of a relation (*relation accuracy*), or of a whole database (*database accuracy*).

When considering accuracy for sets of values instead of single values, a further notion of accuracy can be introduced, namely duplication. *Duplication* occurs when a real-world entity is stored twice or more in a data source. Of course, if a primary key consistency check is performed when populating a relational table, a duplication problem does not occur if the primary key assignment has been made with a reliable procedure. The duplication problem is more relevant for files or other data structures that do not allow the definition of key constraints. A typical cost of duplication is, for example, the additional mailing cost enterprises pay for mailing customers, when customers are stored more than once in the their database. An indirect cost must be added to this direct cost, which consists of the loss of reputation of the enterprise in the eyes of its customers who may be bothered by having to receive the same material more than once.

For relation and database accuracy, for both syntactic and semantic accuracy, a *ratio* is typically calculated between accurate values and the total number of values. For instance, the accuracy of a relation can be measured as the ratio between the number of correct cell values and the total number of cells in the table. More complex metrics can be defined that consider comparison functions; for instance, as we said before, a typical process for syntactic accuracy evaluation is to match tuples from the source under examination with tuples of another source which is supposed to contain the same, but correct tuples.

In such a process, accuracy errors on attribute values can be either those that do not affect the tuple matching, or those that can stop the process itself, not allowing the matching. For instance, an accuracy error on an attribute `SocialSecurityNumber` (`SSN`) value can seriously affect the matching

attempt; instead, given that SSNs are used for matching, an accuracy error on an attribute with a minor identification power, such as `Age`, cannot prevent the identification process from being carried out correctly. In the rest of this section we illustrate a few metrics (see [74]) taking these aspects into account.

Let us consider a relation schema `R` consisting of `K` attributes and a relational table `r` consisting of `N` tuples. Let $q_{ij}$ ($i = 1..N$, $j = 1..K$) be a boolean variable defined to correspond to the cell values $y_{ij}$ such that $q_{ij}$ is equal to 0 if $y_{ij}$ is syntactically accurate, while otherwise it is equal to 1.

In order to identify whether or not accuracy errors affect a matching of a relational table `r` with a reference table `r'` containing correct values, we introduce a further boolean variable $s_i$ equal to 0 if the tuple $\mathtt{t}_i$ matches a tuple in `r'`, and otherwise equal to 1. We can introduce three metrics to distinguish the relative importance of value accuracy in the context of the tuple. The first two metrics have the purpose of giving a different importance to errors on attributes that have a higher identification power, in line with the above discussion.

The first metric is called *weak accuracy error*, and is defined:

$$\sum_{i=1}^{N} \frac{\beta((q_i > 0) \bigwedge (s_i = 0))}{N},$$

where $\beta(.)$ is a boolean variable equal to 1 if the condition in parentheses is true, 0 otherwise, and $q_i = \sum_{j=1}^{K} q_{ij}$. Such metric considers the case in which for a tuple $\mathtt{t}_i$ accuracy errors occur ($q_i > 0$) but do not affect identification ($s_i = 0$).

The second metric is called *strong accuracy error*, and is defined as

$$\sum_{i=1}^{N} \frac{\beta((q_i > 0) \bigwedge (s_i = 1))}{N},$$

where $\beta(.)$ and $q_i$ have the same meaning as above. Such a metric considers the case in which accuracy errors occur ($q_i > 0$) for a tuple $\mathtt{t}_i$ and actually do affect identification ($s_i = 1$).

The third metric gives the percentage of accurate tuples matched with the reference table. It is expressed by the degree of syntactic accuracy of the relational instance `r`

$$\sum_{i=1}^{N} \frac{\beta((q_i = 0) \bigwedge (s_i = 0))}{N}$$

by actually considering the fraction of accurate ($q_i = 0$) matched ($s_i = 0$) tuples.

## 2.2 Completeness

Completeness can be generically defined as "the extent to which data are of sufficient breadth, depth, and scope for the task at hand" [205]. In [161],

three types of completeness are identified. *Schema completeness* is defined as the degree to which concepts and their properties are not missing from the schema. *Column completeness* is defined as a measure of the missing values for a specific property or column in a table. *Population completeness* evaluates missing values with respect to a reference population.

If focusing on a specific data model, a more precise characterization of completeness can be given. In the following we refer to the relational model.

### 2.2.1 Completeness of Relational Data

Intuitively, the completeness of a table characterizes the extent to which the table represents the corresponding real world. Completeness in the relational model can be characterized with respect to: (i) the presence/absence and meaning of null values, and (ii) the validity of one of the two assumptions called *open world assumption* and *closed world assumption*. We now introduce the two issues separately.

In a model *with* null values, the presence of a null value has the general meaning of a missing value, i.e., a value that exists in the real world but for some reason is not available. In order to characterize completeness, it is important to understand *why* the value is missing. Indeed, a value can be missing either because it exists but is unknown, or because it does not exist at all, or because it may exist but it is not actually known whether it exists or not. For a general discussion on the different types of null values see [11]; here we describe the three types of null values, by means of an example.

Let us consider a `Person` relation with the attributes `Name`, `Surname`, `BirthDate`, and `Email`. The relation is shown in Figure 2.2. For the tuples with `Id` equal to 2, 3, and 4, the `Email` value is `NULL`. Let us suppose that the person represented by tuple 2 has no e-mail: no incompleteness case occurs. If the person represented by tuple 3 has an e-mail, but its value is not known then tuple 3 presents an incompleteness. Finally, if it is not known whether the person represented by tuple 4 has an e-mail or not, incompleteness may not be the case.

| ID | Name | Surname | BirthDate | Email |
|---|---|---|---|---|
| 1 | John | Smith | 03/17/1974 | smith@abc.it |
| 2 | Edward | Monroe | 02/03/1967 | NULL |
| 3 | Anthony | White | 01/01/1936 | NULL |
| 4 | Marianne | Collins | 11/20/1955 | NULL |


**Fig. 2.2.** The `Person` relation, with different null value meanings for the e-mail attribute

In logical models for databases, such as the relational model, there are two different assumptions on the completeness of data represented in a relation instance `r`. The *closed world assumption* (CWA) states that only the values actually present in a relational table `r`, and no other values represent facts of the real world. In the *open world assumption* (OWA) we can state neither the truth nor the falsity of facts not represented in the tuples of `r`.

From the four possible combinations emerging from (i) considering or not considering null values, and (ii) OWA and CWA, we will focus on the following two most interesting cases:

1. model without null values with OWA;
2. model with null values with CWA.

In a model *without* null values with OWA, in order to characterize completeness we need to introduce the concept of *reference relation*. Given the relation `r`, the reference relation of `r`, called `ref(r)`, is the relation containing all the tuples that satisfy the relational schema of `r`, i.e., that represent objects of the real world that constitute the present true extension of the schema.

As an example, if `Dept` is a relation representing the employees of a given department, and one specific employee of the department is not represented as a tuple of `Dept`, then the tuple corresponding to the missing employee is in `ref(Dept)`, and `ref(Dept)` differs from `Dept` in exactly that tuple. In practical situations, the reference relations are rarely available. Instead their cardinality is much easier to get. There are also cases in which the reference relation is available but only periodically (e.g., when a census is performed).

On the basis of the reference relation, the completeness of a relation `r` is measured in a model without null values as the fraction of tuples actually represented in the relation `r`, namely, its *size* with respect to the total number of tuples in `ref(r)`:

$$C(r) = \frac{|r|}{|ref(r)|}$$

As an example, let us consider the citizens of Rome. Assume that, from the personal registry of Rome's municipality, the overall number is six million. Let us suppose that a company stores data on Rome's citizens for the purpose of its business; if the cardinality of the relation `r` storing the data is 5,400,000, then $C(r)$ is equal to 0.9.

In the model with null values with CWA, specific definitions for completeness can be provided by considering the granularity of the model elements, i.e., value, tuple, attribute and relations, as shown in Figure 2.3. Specifically, it is possible to define

- a *value completeness*, to capture the presence of null values for some fields of a tuple;
- a *tuple completeness*, to characterize the completeness of a tuple with respect to the values of all its fields;

- an *attribute completeness*, to measure the number of null values of a specific attribute in a relation;
- a *relation completeness*, to capture the presence of null values in a whole relation.

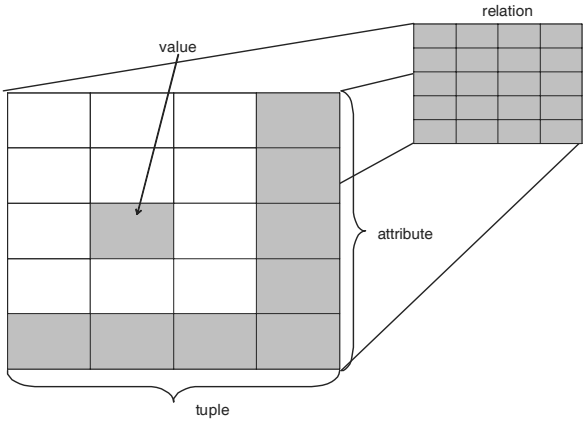



**Fig. 2.3.** Completeness of different elements in the relational model

As an example, in Figure 2.4, a `Student` relation is shown. The tuple completeness evaluates the percentage of specified values in the tuple with respect to the total number of attributes of the tuple itself. Therefore, in the example, the tuple completeness is 1 for tuples 6754 and 8907, 0.8 for tuple 6587, equal to 0.6 for tuple 0987, and so on. One way to see the tuple completeness is as a measure of the information content of the tuple, with respect to its maximum potential information content. With reference to this interpretation, we are implicitly assuming that all values of the tuple contribute equally to the total information content of the tuple. Of course, this may not be the case, as different applications can weight the attributes of a tuple differently.

The attribute completeness evaluates the percentage of specified values in the column corresponding to the attribute with respect to the total number of values that should have been specified. In Figure 2.4, let us consider an application calculating the average of the votes obtained by students. The absence of some values for the `Vote` attribute simply implies a deviation in the calculation of the average; therefore, a characterization of `Vote` completeness may be useful.

The relation completeness is relevant in all applications that need to evaluate the completeness of a whole relation, and can admit the presence of null values on some attributes. Relation completeness measures how much information is represented in the relation by evaluating the content of the information actually available with respect to the maximum possible content,

i.e., without null values. According to this interpretation, completeness of the relation `Student` in Figure 2.4 is 53/60.

| StudentID | Name | Surname | Vote | ExaminationDate |
|---|---|---|---|---|
| 6754 | Mike | Collins | 29 | 07/17/2004 |
| 8907 | Anne | Herbert | 18 | 07/17/2004 |
| 6578 | Julianne | Merrals | NULL | 07/17/2004 |
| 0987 | Robert | Archer | NULL | NULL |
| 1243 | Mark | Taylor | 26 | 09/30/2004 |
| 2134 | Bridget | Abbott | 30 | 09/30/2004 |
| 6784 | John | Miller | 30 | NULL |
| 0098 | Carl | Adams | 25 | 09/30/2004 |
| 1111 | John | Smith | 28 | 09/30/2004 |
| 2564 | Edward | Monroe | NULL | NULL |
| 8976 | Anthony | White | 21 | NULL |
| 8973 | Marianne | Collins | 30 | 10/15/2004 |

**Fig. 2.4.** `Student` relation exemplifying the completeness of tuples, attributes and relations.

### 2.2.2 Completeness of Web Data

Data that are published in Web information systems can be characterized by evolution in time. While in the traditional paper-based media, information is published once and for all, Web information systems are characterized by information that is continuously published.

Let us consider the Web site of a university, where a list of courses given at that university in the current academic year is published. At a given moment, the list can be considered *complete* in the sense that it includes all the courses that have been officially approved. Nevertheless, it is also known that more courses will be added to the list, pending their approval. Therefore, there is the need to apprehend how the list will evolve in time with respect to completeness. The *traditional* completeness dimension provides only a static characterization of completeness. In order to consider the temporal dynamics of completeness, as needed in Web information systems, we introduce the notion of completability.

We consider a function $C(t)$, defined as the value of completeness at the instant $t$, with $t \in [\texttt{t\_pub}, \texttt{t\_max}]$, where `t_pub` is the initial instant of publication of data and `t_max` corresponds to the maximum time within which the series of the different scheduled updates will be completed. Starting from the function $C(t)$, we can define the *completability* of the published data as

$$\int_{t\_curr}^{t\_max} C(t),$$

where t_curr is the time at which completability is evaluated and t_curr < t_max.

Completability, as shown in Figure 2.5, can be graphically depicted as an area Cb of a function that represents how completeness evolves between an instant t_curr of observation and t_max. Observe that the value corresponding to t_curr is indicated as c_curr; c_max is the value for completeness estimated for t_max. The value c_max is a real reachable limit that can be specified for the completeness of the series of elements; if this real limit does not exist, c_max is equal to 1. In Figure 2.5, a reference area A is also shown, defined as

$$(t\_max - t\_curr) * \frac{c\_max - c\_pub}{2},$$

that, by comparison with Cb, allows us to define ranges [High, Medium, Low] for completability.

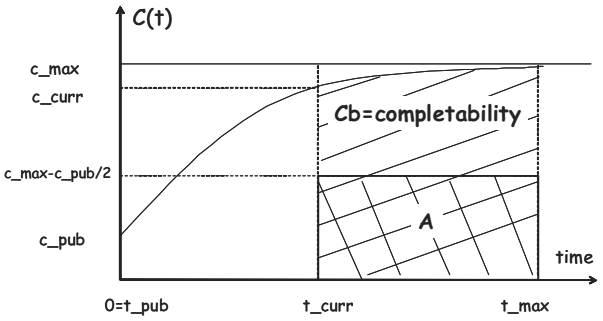



**Fig. 2.5.** A graphical representation of completability

With respect to the example above, considering the list of courses published on a university Web site, the completeness dimension gives information about the current degree of completeness; the completability information gives the information about how fast this degree will grow in time, i.e., how fast the list of courses will be completed. The interested reader can find further details in [159].

## 2.3 Time-Related Dimensions: Currency, Timeliness, and Volatility

An important aspect of data is their change and update in time. In Chapter 1 we provided a classification of types of data according to the temporal dimension, in terms of stable, long-term-changing, and frequently changing data. The principal time-related dimensions proposed for characterizing the above three types of data are currency, volatility, and timeliness.

*Currency* concerns how promptly data are updated. As an example in Figure 2.1, the attribute `#Remakes` of movie 4 has low currency because a remake of movie 4 has been done, but this information did not result in an increased value for the number of remakes. Similarly, if the residential address of a person is updated, i.e., it corresponds to the address where the person lives, then the currency is high.

*Volatility* characterizes the frequency with which data vary in time. For instance, stable data such as birth dates have volatility equal to 0, as they do not vary at all. Conversely, stock quotes, a kind of frequently changing data, have a high degree of volatility due to the fact that they remain valid for very short time intervals.

*Timeliness* expresses how current data are for the task at hand. The timeliness dimension is motivated by the fact that it is possible to have current data that are actually useless because they are *late* for a specific usage. For instance, the timetable for university courses can be current by containing the most recent data, but it cannot be timely if it is available only after the start of the classes.

We now provide possible metrics of time-related dimensions. Currency can be typically measured with respect to *last update* metadata, which correspond to the last time the specific data were updated. For data types that change with a fixed frequency, last update metadata allow us to compute currency straightforwardly. Conversely, for data types whose change frequency can vary, one possibility is to calculate an average change frequency and perform the currency computation with respect to it, admitting errors. As an example, if a data source stores residence addresses that are estimated to change every five years, then an address with its last update metadata reporting a date corresponding to one month before the observation time can be assumed to be *current*; in contrast, if the date reported is ten years before the observation time, it is assumed to be *not current*.

Volatility is a dimension that inherently characterizes certain types of data. A metric for volatility is given by the length of time (or its inverse) that data remain valid.

Timeliness implies that data not only are current, but are also in time for events that correspond to their usage. Therefore, a possible measurement consists of (i) a currency measurement and (ii) a check that data are available *before* the planned usage time.

More complex metrics can be defined for time-related dimensions. As an example, we cite the metric defined in [17], in which the three dimensions currency, volatility, and timeliness are linked by defining timeliness as a function of currency and volatility. More specifically,

1. Currency is defined as

$$Currency = Age + (DeliveryTime - InputTime),$$

where $Age$ measures how old the data unit is when received, $DeliveryTime$ is the time the information product is delivered to the

customer, and $InputTime$ is the time the data unit is obtained. Therefore, currency is the sum of how old data are when received ($Age$), plus a second term that measures how long data have been in the information system, ($DeliveryTime - InputTime$);
2. Volatility is defined as the length of time data remains valid;
3. Timeliness is defined as,

$$\max\{0, 1 - \frac{currency}{volatility}\}.$$

Timeliness ranges from 0 to 1, where 0 means bad timeliness and 1 means good timeliness.

Observe that the relevance of currency depends on volatility: data that are highly volatile must be current, while currency is less important for data with low volatility.

## 2.4 Consistency

The consistency dimension captures the violation of semantic rules defined over (a set of) data items, where items can be tuples of relational tables or records in a file. With reference to relational theory, *integrity constraints* are an instantiation of such semantic rules. In statistics, *data edits* are another example of semantic rules that allow for the checking of consistency.

### 2.4.1 Integrity Constraints

The interested reader can find a detailed discussion on integrity constraints in the relational model in [11]. The purpose of this section is to summarize the main concepts, useful for introducing the reader to consistency-related topics.

Integrity constraints are properties that must be satisfied by all instances of a database schema. Although integrity constraints are typically defined on schemas, they can at the same time be checked on a specific instance of the schema that presently represents the extension of the database. Therefore, we may define integrity constraints for schemas, describing a schema quality dimension, and for instances, representing a data dimension. In this section, we will define them for instances, while in section 2.7 we will define them for schemas.

It is possible to distinguish two main categories of integrity constraints, namely, *intrarelation constraints* and *interrelation constraints*. Intrarelation integrity constraints can regard single attributes (also called *domain constraints*) or multiple attributes of a relation.

Let us consider an `Employee` relation schema, with the attributes `Name`, `Surname`, `Age`, `WorkingYears`, and `Salary`. An example of the domain constraint defined on the schema is "`Age` is included between 0 and 120." An

example of a multiple attribute integrity constraint is: "If `WorkingYears` is less than 3, than `Salary` could not be more than 25.000 Euros per year."

Interrelation integrity constraints involve attributes of more than one relation. As an example, consider the `Movies` relation instance in Figure 2.1. Let us consider another relation, `OscarAwards`, specifying the Oscar awards won by each movie, and including an attribute `Year` corresponding to the year when the award was assigned. An example of interrelation constraint states that " `Year` of the `Movies` relation must be equal to `Year` of `OscarAwards`."

Most of the considered integrity constraints are *dependencies*. The following main types of dependencies can be considered:

- *Key Dependency.* This is the simplest type of dependency. Given a relation instance `r`, defined over a set of attributes, we say that for a subset `K` of the attributes a key dependency holds in `r`, if no two rows of `r` have the same `K`-values. For instance, an attribute like `SocialSecurityNumber` can serve as a key in any relation instance of a relation schema `Person`. When key dependency constraints are enforced, no duplication will occur within the relation (see also Section 2.1 on duplication issues).
- *Inclusion Dependency.* Inclusion dependency is a very common type of constraint, and is also known as *referential constraint*. An inclusion dependency over a relational instance `r` states that some columns of `r` are contained in other columns of `r` or in the instances of another relational instance `s`. A *foreign key constraint* is an example of inclusion dependency, stating that the referring columns in one relation must be contained in the primary key columns of the referenced relation.
- *Functional Dependency.* Given a relational instance `r`, let `X` and `Y` be two nonempty sets of attributes in `r`. `r` satisfies the functional dependency $\mathtt{X} \rightarrow \mathtt{Y}$, if the following holds for every pair of tuples $\mathtt{t}_1$ and $\mathtt{t}_2$ in `r`:

  $$\text{If } \mathtt{t}_1.\mathtt{X} = \mathtt{t}_2.\mathtt{X}, \text{ then } \mathtt{t}_1.\mathtt{Y} = \mathtt{t}_2.\mathtt{Y},$$

  where the notation $\mathtt{t}_1.\mathtt{X}$ means the projection of the tuple $\mathtt{t}_1$ onto the attributes in `X`. In Figure 2.6, examples of relations respectively satisfying and violating a functional dependency $\mathtt{AB} \rightarrow \mathtt{C}$ are shown. In the figure, the relation $\mathtt{r}_1$ satisfies the functional dependency, as the first two tuples, having the same values for the attribute `A` and the attribute `B`, also have the same value for the attribute `C`. The relation $\mathtt{r}_2$ does not satisfy the functional dependency, since the first two tuples have a different `C` field.

### 2.4.2 Data Edits

In the previous section, integrity constraints were discussed within the relational model as a specific category of consistency semantic rules. However, where data are not relational, consistency rules can still be defined. As an example, in the statistical field, data coming from census questionnaires have

| A | B | C | D |
|---|---|---|---|
| $a_1$ | $b_1$ | $c_1$ | $d_1$ |
| $a_1$ | $b_1$ | $c_1$ | $d_2$ |
| $a_1$ | $b_2$ | $c_3$ | $d_3$ |

$r_1$

| A | B | C | D |
|---|---|---|---|
| $a_1$ | $b_1$ | $c_2$ | $d_1$ |
| $a_1$ | $b_1$ | $c_1$ | $d_2$ |
| $a_1$ | $b_2$ | $c_3$ | $d_3$ |

$r_2$

**Fig. 2.6.** Example of functional dependencies

a structure corresponding to the *questionnaire schema*. The semantic rules are thus defined over such a structure in a way very similar to relational constraints. Such rules, called *edits*, are less powerful than integrity constraints because they do not rely on a data model like the relational one. Nevertheless, data editing has been done extensively in the national statistical agencies since the 1950s, and has revealed a fruitful and effective area of application. *Data editing* is defined as the task of detecting inconsistencies by formulating rules that must be respected by every correct set of answers. Such rules are expressed as *edits*, which denote error conditions.

As an example, an inconsistent answer to a questionnaire can be to declare

`marital status = ‘‘married’’, age = ‘‘5 years old’’`

The rule to detect this kind of errors could be the following:

`if marital status is married, age must not be less than 14.`

The rule can be put in the form of an edit, which expresses the error condition, namely,

$$\texttt{marital status} = \texttt{married} \wedge \texttt{age} < 14$$

After the detection of erroneous records, the act of correcting erroneous fields by restoring correct values is called *imputation*. The problem of localizing errors by means of edits and imputing erroneous fields is known as the *edit-imputation problem*.In Chapter 4 we will examine some issues and methods for the edit-imputation problem.

## 2.5 Other Data Quality Dimensions

In the previous section, a description of the principal data quality dimensions was provided. However, in the data quality literature, several further dimensions have been proposed in addition to the four described ones.

There are general proposals for sets of dimensions that aim to fully specify the data quality concept in a general setting (see Section 2.6). Some other proposals are related to specific domains that need ad hoc dimensions in order to capture the peculiarities of the domain. For instance, specific data quality dimensions are proposed in the following domains:

1. The *archival domain* (see [217] and [111]) and the Interpares project [101], which makes use of dimensions such as *condition* (of a document) that refers to the physical suitability of the document for scanning.
2. The *statistical domain*; every National bureau of census and international organizations such as the European Union or the International Monetary Fund define several dimensions for statistical and scientific data (see [96]), such as *integrity*, on the notion that statistical systems should be based on adherence to the principle of objectivity in the collection, compilation, and dissemination of statistics.
3. The *geographical and geospatial domain* (see [152], [89], and [101]), where the following dimensions are proposed: (i) *positional accuracy*, defined as a quality parameter indicating the accuracy of geographical positions, and (ii) *attribute/thematic accuracy*, defined as the positional and/or value accuracy of properties such as sociodemographic attributes in thematic maps.

In the following we will describe some new dimensions that are gaining increasing importance in networked information systems. With the advent of Web information systems, and peer-to-peer information systems, the number of sources of data increases dramatically, and provenance on available data is difficult to evaluate in the majority of cases. This is a radical change from old, centralized systems (still widespread in some organizations, such as banks), where data sources and data flows are accurately controlled and monitored. In this context, new quality dimensions arise; among them we now discuss interpretability, synchronization in time series, and, in more detail, accessibility and (the set of) dimensions proposed for characterizing the quality of an information source. Other dimensions are introduced and discussed in [50].

*Interpretability* concerns the documentation and metadata that are available to correctly interpret the meaning and properties of data sources. In order to maximize interpretability, the following types of documentation should be available:

1. the conceptual schema of the file(s) or database(s) made available;
2. the integrity constraints that hold among data;
3. a set of metadata for cross-domain information resource description, such as the one described in the standard Dublin core (see [63] for an exhaustive introduction to this standard including, among others, metadata like *creator*, *subject*, *description*, *publisher*, *date*, *format*, *source*, and *language*);
4. a certificate describing available measures of data quality dimensions and schema dimensions; and
5. information on the history and provenance of data, i.e., how and where it has been created, produced, and maintained. For a discussion on provenance of data, see Chapter 3.

*Synchronization between different time series* concerns proper integration of data having different time stamps. Synchronization is a major problem for

organizations that produce statistics, and in which data come from different sources of collected data with different time stamps. For example, if in a company we are combining data on expenditures and data on revenues, it is important to get the data synchronized correctly, otherwise the analysis could produce incorrect results. Statistical methods, the discussion of which is out of the scope of this book, are used to synchronize the data and allow their fusion. We refer again to [50] for more details on this point.

### 2.5.1 Accessibility

Publishing large amounts of data in Web sites is not a sufficient condition for its availability to everyone. In order to access it, a user needs to access a network, to understand the language to be used for navigating and querying the Web, and to perceive with his or her senses the information made available. *Accessibility* measures the ability of the user to access the data from his or her own culture, physical status/functions, and technologies available. We focus in the following on causes that can reduce physical or sensorial abilities, and, consequently, can reduce accessibility, and we briefly outline corresponding guidelines to achieve accessibility. Among others, the World Wide Web Consortium [198] defines the individuals with disabilities as subjects that,

1. may not be able to see, hear, move, or process some types of information easily or at all;
2. may have difficulty reading or comprehending text;
3. may not have to or be able to use a keyboard or mouse;
4. may have a text-only screen, a small screen, or a slow Internet connection;
5. may not speak or understand a natural language fluently.

Several guidelines are provided by international and national bodies to govern the production of data, applications, services, and Web sites in order to guarantee accessibility. In the following, we describe some guidelines related to data provided by the World Wide Web Consortium in [198].

The first, and perhaps most important, guideline addresses providing equivalent alternatives to auditory and visual content, called *text equivalent content*. In order for a text equivalent to make an image accessible, the text content can be presented to the user as synthesized speech, braille, and visually displayed text. Each of these three mechanisms uses a different sense, making the information accessible to groups affected by a variety of sensory and other disabilities. In order to be useful, the text must convey the same function or purpose as the image. For example, consider a text equivalent for a photographic image of the continent of Africa as seen from a satellite. If the purpose of the image is mostly that of decoration, then the text "Photograph of Africa as seen from a satellite" might fulfill the necessary function. If the purpose of the photograph is to illustrate specific information about African geography, such as its organization and subdivision into states, then the text equivalent should convey that information with more articulate and

informative text. If the photograph has been designed to allow the user to select the image or part of it (e.g., by clicking on it) for information about Africa, equivalent text could be "Information about Africa", with a list of items describing the parts that can be selected. Therefore, if the text conveys the same function or purpose for the user with a disability as the image does for other users, it can be considered a text equivalent.

Other guidelines suggest

- avoiding the use of color as the only means to express semantics, helping daltonic people appreciate the meaning of data;
- usage of clear natural language, by providing expansions of acronyms, improving readability, a frequent use of plain terms;
- designing a Web site that ensures device independence using features that enable activation of page elements via a variety of input devices;
- providing context and orientation information to help users understand complex pages or elements.

Several countries have enacted specific laws to enforce accessibility in public and private Web sites and applications used by citizens and employees in order to provide them effective access and reduce the digital divide.

### 2.5.2 Quality of Information Sources

Several dimensions have been proposed for characterizing the quality of an information source as a whole.

In Wang and Strong [205], three dimensions model how "trustable" is the information source providing the data. These dimensions are believability, reputation, and objectivity. Believability considers whether a certain source provides data that can be regarded as true, real and credible. Reputation considers how trustable is the information source. Objectivity takes into account impartiality of sources in data provisioning.

Similarly to the above described dimensions, reliability (or credibility) is also proposed as a dimension for representing whether a source provides data conveying the right information (e.g., in Wand and Wang [199]).

More recently, with the increasing interest in peer-to-peer systems, the quality characterization of a source (or peer) is gaining importance. Indeed, in such systems that are completely open, there is the need to assess and filter data that circulate in the system, and one possibility is to rely on the trustability of each peer. As an example, in [59], a trust model for information peers is proposed, in which a trust level is associated to a certain peer for each typology of data provided to the community. The interested reader can find more details on trust issues in peer-to-peer systems in Chapter 9.

## 2.6 Approaches to the Definition of Data Quality Dimensions

In this section we focus on the general proposals for dimensions by illustrating some of them. There are three main approaches adopted for proposing comprehensive sets of the dimension definitions, namely, theoretical, empirical, and intuitive. The *theoretical approach* adopts a formal model in order to define or justify the dimensions. The *empirical approach* constructs the set of dimensions starting from experiments, interviews, and questionnaires. The *intuitive approach* simply defines dimensions according to common sense and practical experience.

In the following, we summarize three main proposals that clearly represent the approaches to dimension definitions: Wand and Wang [199], Wang and Strong [205], and Redman [167].

### 2.6.1 Theoretical Approach

A theoretical approach to the definition of data quality is proposed in Wand and Wang [199]. This approach considers an information system (IS) as a representation of a *real-world system* (RW); RW is *properly represented* in an IS if (i) there exists an exhaustive mapping $RW \rightarrow IS$, and (ii) no two states in RW are mapped into the same state in an IS, i.e., the inverse mapping is a function (see Figure 2.7).

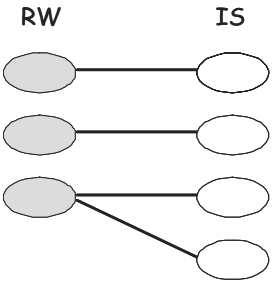
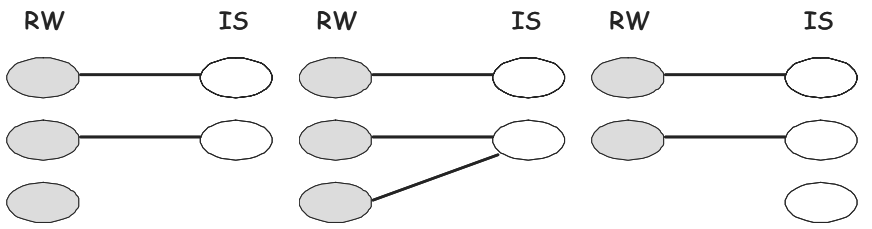



**Fig. 2.7.** Proper representation of the real world system in the theoretical approach from [199]

All deviations from proper representations generate deficiencies. They distinguish between *design deficiencies* and *operation deficiencies*. Design deficiencies are of three types: *incomplete representation, ambiguous representation,* and *meaningless states.* They are graphically represented in Figure 2.8.

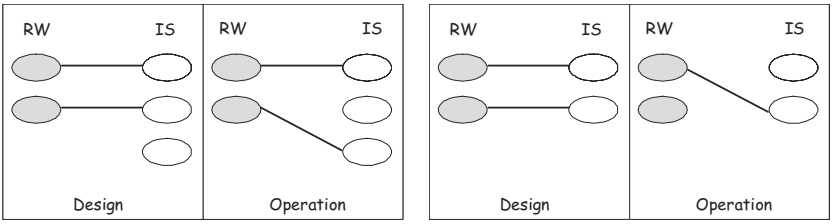
Only one type of operation deficiency is identified, in which a state in RW might be mapped to a wrong state in an IS; this is referred to as *garbling.*

Garbling with a map to a meaningless state is dangerous, as it will preclude a map back to a real world state (see Figure 2.9a). Garbling to a meaningful but wrong state will allow the user to map back to a real world state (see Figure 2.9b).




(a) Incomplete (b) Ambiguous (c) Meaningless

**Fig. 2.8.** Incomplete, ambiguous, and meaningless representations of the real world system in the theoretical approach




(a) Not meaningful (b) Meaningful

**Fig. 2.9.** Garbling representations of the real world system from [199]

A set of data quality dimensions are defined by making references to described deficiencies. More specifically, the identified dimensions are (the quoted text is from [199])

- *Accuracy*: "inaccuracy implies that the information system represents a real world state different from the one that should have been represented." Inaccuracy refers to a garbled mapping into a wrong state of the IS, where it is possible to infer a valid state of the real world though not the correct one (see Figure 2.9b).

- *Reliability* indicates "whether the data can be counted on to convey the right information; it can be viewed as correctness of data." No interpretation in terms of data deficiencies is given.
- *Timeliness* refers to "the delay between a change of the real-world state and the resulting modification of the information system state." Lack of timeliness may lead to an IS state that reflects a past RW state.
- *Completeness* is "the ability of an information system to represent every meaningful state of the represented real world system." Completeness is of course tied to incomplete representations.
- *Consistency* of data values occurs if there is more than one state of the information system matching a state of the real-world system; therefore "inconsistency would mean that the representation mapping is one-to-many." This is captured by representation, so the inconsistency is not considered a result of a deficiency.

| Category | Dimension | Definition: the extent to which ... |
|---|---|---|
| Intrinsic | Beleivability | data are accepted or regardedas true, real and credible |
| | Accuracy | data are correct, reliable and certified free of error |
| | Objectivity | data are unbiased and impartial |
| | Reputation | data are trusted or highly regarded in terms of their source and content |
| Contextual | Value-added | data are beneficial and provide advantages for their use |
| | Relevancy | data are applicable and useful for the task at hand |
| | Timeliness | the age of the data is appropriate for the task at hand |
| | Completeness | data are of sufficient depth, breadth, and scope for the task at hand |
| | Appropriate amount of data | the quantity or volume of available data is appropriate |
| Representational | Intepretability | data are in appropriate language and unit and the data definitions are clear |
| | Ease of understanding | data are clear without ambiguity and easily comprehended |
| | Representational consistency | data are always presented in the same format and are compatible with the previous data |
| | Concise representation | data are compactly represented without behing overwhelmed |
| Accessibility | Accessibility | data are available or easily and quickly retrieved |
| | Access security | access to data can be restricted and hence kept secure |

**Fig. 2.10.** Dimensions proposed in the empirical approach

### 2.6.2 Empirical Approach

In the proposal discussed in Wang and Strong [205], data quality dimensions have been selected by interviewing data consumers. Starting from 179 data quality dimensions, the authors selected 15 different dimensions, represented

in Figure 2.10 with their definitions. A two-level classification is proposed, in which each of four *categories* is further specialized into a number of *dimensions*. The four categories are

- *intrinsic data quality*, capturing the quality that data has on its own. As an example, accuracy is a quality dimension that is intrinsic to data;
- *contextual data quality* considers the context where data are used. As an example, the completeness dimension is strictly related to the context of the task;
- *representational data quality* captures aspects related to the quality of data representation, e.g., interpretability;
- *accessibility data quality* is related to the accessibility of data and to a further non-functional property of data access, namely, the level of security.

### 2.6.3 Intuitive Approach

Redman [167] classifies data quality dimensions according to three categories, namely, conceptual schema, data values, and data format. *Conceptual schema* dimensions correspond to what we called schema dimensions. *Data value* dimensions refer specifically to values, independently of the internal representation of data; this last aspect is covered by *data format* dimensions. Our focus here is on data extension; therefore, in Figure 2.11, we provide the definitions for data value and format dimensions only.

### 2.6.4 A Comparative Analysis of the Dimension Definitions

According to the definitions described in the previous section, there is no general agreement either on which set of dimensions defines data quality or on the exact meaning of each dimension. In fact, in the illustrated proposals, dimensions are not defined in a measurable and formal way. Instead, they are defined by means of descriptive sentences in which the semantics are consequently disputable. Nevertheless, we attempt to make a comparison between the different definitions provided with the purpose of showing possible agreements and disagreements in the different proposals. In order to cover a larger number of proposals, besides the previously described Wand and Wang [199], Wang and Strong [205], and Redman [167], we also consider Jarke et al. [104], Bovee et al. [31], Naumann [139], and Liu [120]. Hereafter we will refer to the proposals with the name of the first author of the work.

With regard to time-related dimensions, in Figure 2.12, definitions for currency, volatility, and timeliness by different authors are illustrated. In the figure, Wand and Redman provide very similar definitions but for different dimensions, i.e., for timeliness and currency, respectively. Wang and Liu assume the same meaning for timeliness, Naumann proposes a very different definition for it, and Bovee only provides a definition for timeliness in terms of currency and volatility. Bovee's currency corresponds to timeliness as defined

| Dimension Name | Type of dimension | Definition |
|---|---|---|
| Accuracy | data value | Distance between v and v', considered as correct |
| Completeness | data value | Degree to which values are present in a data collection |
| Currency | data value | Degree to which a datum is up-to-date |
| Consistency | data value | Coherence of the same datum, represented in multiple copies, or different data to respect integrity constraints and rules |
| Appropriateness | data format | One format is more appropriate than another if it is more suited to user needs |
| Interpretability | data format | Ability of the user to interpret correctly values from their format |
| Portability | data format | The format can be applied to as a wide set of situations as possible |
| Format precision | data format | Ability to distinguish between elements in the domain that must be distinguished by users |
| Format flexibility | data format | Changes in user needs and recording medium can be easily accommodated |
| Ability to represent null values | data format | Ability to distinguish neatly (without ambiguities) null and default values from applicable values of the domain |
| Efficient use of memory | data format | Efficiency in the physical representation. An icon is less efficient than a code |
| Representation consistency | data format | Coherence of physical instances of data with their formats |

**Fig. 2.11.** Dimensions proposed in the intuitive approach [167]

by Wang and Liu. Volatility has a similar meaning in Bovee and Jarke. The comparison shows that there is no substantial agreement on the names to use for time-related dimensions; indeed, currency and timeliness are often used to refer to the same concept. There is not even agreement on the semantics of a specific dimension; indeed, for timeliness, different meanings are provided by different authors.

With regard to completeness, in Figure 2.13, different proposals for completeness definitions are shown. By comparing such definitions, it emerges that there is substantial agreement on what completeness is, although it often refers to different granularity levels and different data model elements, e.g., information system in Wand, data warehouse in Jarke, and entity in Bovee.

### 2.6.5 Trade-offs Between Dimensions

Data quality dimensions are not independent, i.e., correlations exist between them. If one dimension is considered more important than the others for a specific application, then the choice of favoring it may imply negative consequences for the other ones. In this section, we provide some examples of possible trade-offs.

First, trade-offs may need to be made between timeliness and any one of the three dimensions: accuracy, completeness, and consistency. Indeed, hav-

| Reference | Definition |
|---|---|
| Wand 1996 | Timeliness refers only to the delay between a change of a real world state and the resulting modification of the information system state |
| Wang 1996 | Timeliness is the extent to which age of the data is appropriate for the task at hand |
| Redman 1996 | Currency is the degree to which a datum is up-to-date. A datum value is up-to-date if it is correct in spite of possible discrepancies caused by time-related changes to the correct value |
| Jarke 1999 | Currency describes when the information was entered in the sources and/or the data warehouse.<br>Volatility describes the time period for which information is valid in the real world |
| Bovee 2001 | Timeliness has two componenents: age and volatility. Age or currency is a measure of how old the information is, based on how long ago it was recorded. Volatility is a measure of information instability-the frequency of change of the value for an entity attribute |
| Naumann 2002 | Timeliness is the average age of the data in a source |
| Liu 2002 | Timeliness is the extent to which data are sufficiently up-to-date for a task |

**Fig. 2.12.** Time-related dimensions definitions

| Reference | Definition |
|---|---|
| Wand 1996 | The ability of an information system to represent every meaningful state of the represented real world system. |
| Wang 1996 | The extent to which data are of sufficient breadth, depth and scope for the task at hand |
| Redman 1996 | The degree to which values are present in a data collection |
| Jarke 1999 | Percentage of the real-world information entered in the sources and/or the data warehouse |
| Bovee 2001 | Deals with information having all required parts of an entity's information present |
| Naumann 2002 | It is the quotient of the number of non-null values in a source and the size of the universal relation |
| Liu 2002 | All values that are supposed to be collected as per a collection theory |

**Fig. 2.13.** Completeness dimensions definitions

ing accurate (or complete or consistent) data, may need checks and activities that require time, and thus timeliness is negatively affected. Conversely, having timely data may cause lower accuracy (or completeness or consistency). A typical situation in which timeliness can be preferred to accuracy, completeness, or consistency is given by most Web applications: as the time constraints are often very stringent for Web data, it is possible that such data are deficient with respect to other quality dimensions. For instance, a list of courses published on a university Web site must be timely though there could be accuracy or consistency errors and some fields specifying courses could be missing.

Conversely, when considering an administrative application, accuracy, consistency, and completeness requirements are more stringent than timeliness, and therefore delays are mostly admitted in dimensions other than timeliness.

Another significant case of trade-off is between consistency and completeness [15]. Here the question is "Is it better to have less but consistent data, i.e., poor completeness, or to have more but inconsistent data?". This choice is again very domain specific. As an example, statistical data analysis typically requires a significant and representative amount of data in order to perform the analysis; in this case, the approach is to favor completeness, tolerating inconsistencies, or adopting techniques to solve them. Conversely, when considering the publishing of a list of votes obtained by students as the result of an exam, it is more important to have a list of consistency checked votes than a complete one, possibly deferring the publication of the complete list.

## 2.7 Schema Quality Dimensions

In the previous sections, we provided an in-depth characterization of data quality dimensions. In this section, the focus is on schema quality dimensions. However, there is a strict relationship between quality of schemas and quality of data, as highlighted in the next example. Let us suppose we want to model residence addresses of people; in Figure 2.14, there are two possibilities to model such a concept. Specifically, in Figure 2.14a, the residence addresses are modeled as attributes of a relation `Person`, while in Figure 2.14b, the residence addresses are modeled as a relation `Address`, with the fields `Id`, `StreetPrefix`, `StreetName`, `Number`, `City`, and a relation `ResidenceAddress` storing the address at which the person lives. The solution in Figure 2.14a has some problems. First, representing addresses as a single field creates ambiguity on the meaning of the different components; for instance, in tuple 3 of the `Person` relation, is `4` a civic number or the number of the avenue (it is actually part of the name of the square)? Second, the values of the attribute `Address` can also contain information that is not explicitly required to be represented (e.g., the floor number and zip code of tuples 1 and 2 of the `Person` relation). Third, as the `Person` relation is not normalized, a redundancy problem occurs and hence further errors on the `Address` attribute may be potentially introduced (see the same address values for tuples 1 and 2 of the `Person` relation). On the other hand, the solution in Figure 2.14b is more complex. In real implementation there is often the need to manage trade-offs between the two modeling solutions.

A comprehensive proposal on schema dimensions is described in the book of Redman [167], and includes six dimensions and 15 subdimensions referring to schema quality. Here, we focus on seven subdimensions, which we call dimensions in the following section. In the definitions we are going to provide, we assume that the database schema is the result of the translation of a set of requirements, expressed usually in natural language, into a set of concep-

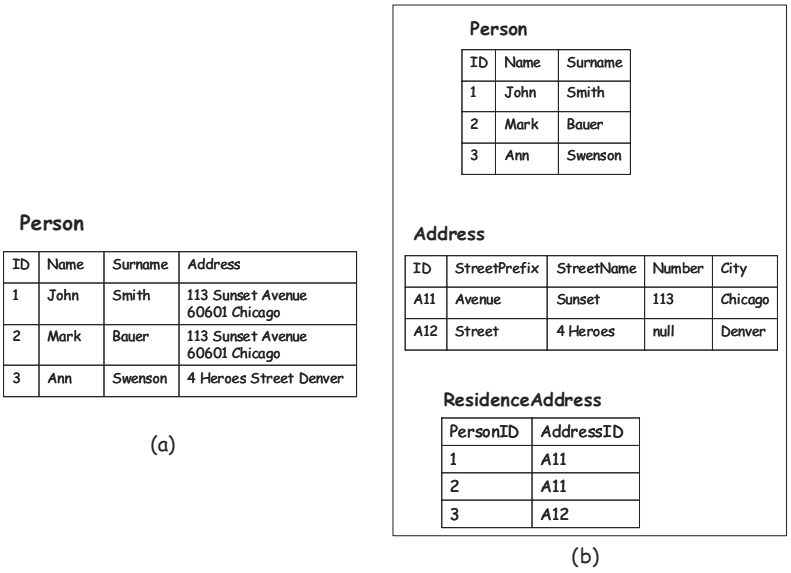



**Fig. 2.14.** Two ways of modeling residence addresses

tual (or logical) structures, expressed in terms of a conceptual (or logical) database model. Two of these dimensions, namely, *readability* and *normalization*, will be discussed in specific sections. We briefly introduce the remaining five dimensions.

1. *Correctness with respect to the model* concerns the correct use of the categories of the model in representing requirements. As an example, in the Entity Relationship model we may represent the logical link between persons and their first names using the two entities `Person` and `FirstName` and a relationship between them. The schema is not correct wrt the model since an entity should be used only when the concept has a unique existence in the real world and has an identifier; this is not the case with `FirstName`, which would be properly represented as an attribute of the entity `Person`.
2. *Correctness with respect to requirements* concerns the correct representation of the requirements in terms of the model categories. Assume that in an organization each department is headed by exactly one manager and each manager may head exactly one department. If we represent `Manager` and `Department` as entities, the relationship between them should be one-to-one; in this case, the schema is correct wrt requirements. If we use a one-to-many relationship, the schema is incorrect.
3. *Minimalization.* A schema is minimal if every part of the requirements is represented only once in the schema. In other words, it is not possible to eliminate some element from the schema without compromising the in-

formation content. Consider the schema in Figure 2.15, which represents several relationships between concepts `Student`, `Course`, and `Instructor`. We represent also minimum and maximum cardinalities of entities in relationships, except in one case, where we indicate the maximum cardinality with the symbol "?". The schema is redundant in the case in which the direct relationship `Assigned to` between `Student` and `Instructor` has the same meaning as the logical composition of the two relationships `Attends` and `Teaches`; otherwise, it is nonredundant. Notice that the schema can be redundant only in the case in which the unspecified maximum cardinality of the entity `Course` is "1", since only in this case does a unique instructor correspond to each course, and the composition of the two relationships `Attends` and `Teaches` may provide the same result as the relationship `Assigned to`.

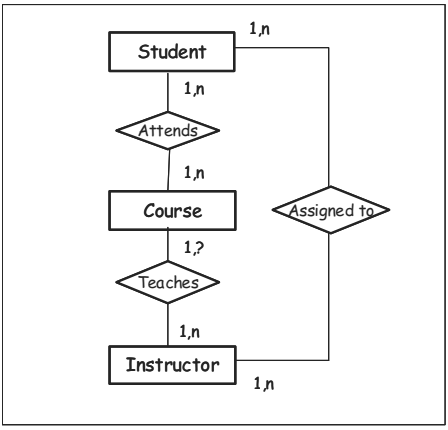



**Fig. 2.15.** A possibly redundant schema

4. *Completeness* measures the extent to which a conceptual schema includes all the conceptual elements necessary to meet some specified requirements. It is possible that the designer has not included certain characteristics present in the requirements in the schema, e.g., attributes related to an entity `Person`; in this case, the schema is incomplete.
5. *Pertinence* measures how many unnecessary conceptual elements are included in the conceptual schema. In the case of a schema that is not pertinent, the designer has gone too far in modeling the requirements, and has included too many concepts.

Completeness and pertinence are two faces of the same issue, i.e., obtaining a schema at the end of the conceptual design phase that is the *exact* correspondence in the model of the reality described by requirements.

### 2.7.1 Readability

Intuitively, a schema is readable whenever it represents the meaning of the reality represented by the schema in a clear way for its intended use. This simple, qualitative definition is not easy to translate in a more formal way, since the evaluation expressed by the word *clearly* conveys some elements of subjectivity. In models, such as the Entity Relationship model, that provide a graphical representation of the schema, called *diagram*, readability concerns both the diagram and the schema itself. We now discuss them.

With regard to the diagrammatic representation, readability can be expressed by a number of *aesthetic criteria* that human beings adopt in drawing diagrams: crossings between lines should be avoided as far as possible, graphic symbols should be embedded in a grid, lines should be made of horizontal or vertical segments, the number of bends in lines should be minimized, the total area of the diagram should be minimized, and, finally, hierarchical structures such as generalization hierarchies among, say, an entity `E1` and two entities `E2` and `E3` should be such that `E1` is positioned at a higher level in the diagram in respect to `E2` and `E3`. Finally, the children entities in the generalization hierarchy should be symmetrical with respect to the parent entity. For further discussion on aesthetic criteria, see [22], and [186].

The above criteria are not respected in the case of the Entity Relationship diagram of Figure 2.16. We can see in the diagram many crossings between lines. Most objects are placed casually in the area of the schema, and it is difficult to identify the group of entities related by generalization hierarchy. The schema, in a few words, has a "spaghetti style."

Following the aesthetic rules described above, we may completely restructure the diagram, leading to the new diagram shown in Figure 2.17. Here, most relevant concepts have a larger dimension, there are no bends in lines, and the generalization hierarchy is more apparent.

The second issue addressed by readability is the simplicity of schema representation. Among the different conceptual schemas that equivalently represent a certain reality, we prefer the one or the ones that are more compact, because compactness favors readability. As an example, in the left hand side of Figure 2.18, we see a schema where the represented entity `City` is related to the three children entities in the generalization hierarchy. Due to the inheritance property [66], which states that all concepts related to the parent entity are also related to all the children entities, we can drop the three occurrences of relationships involving the entity `City` and change them into a single relationship with the entity `Employee`, resulting in a more compact and readable schema.

### 2.7.2 Normalization

The property of *normalization* has been deeply investigated, especially in the relational model, although it expresses a model-independent, general property of schemas.

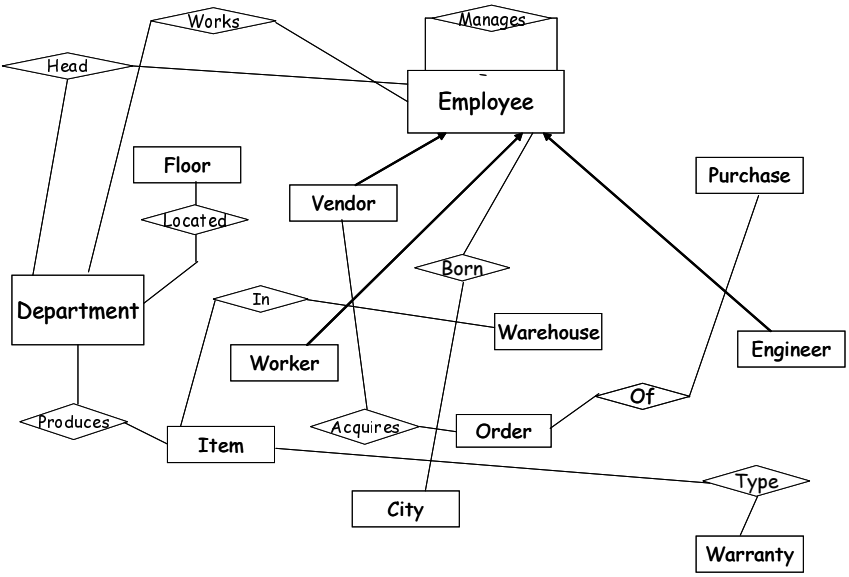



**Fig. 2.16.** "Spaghetti style" Entity Relationship schema

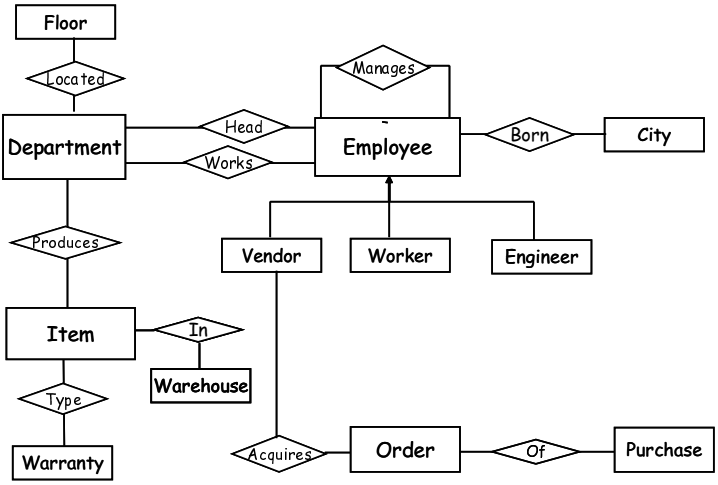



**Fig. 2.17.** An equivalent readable schema

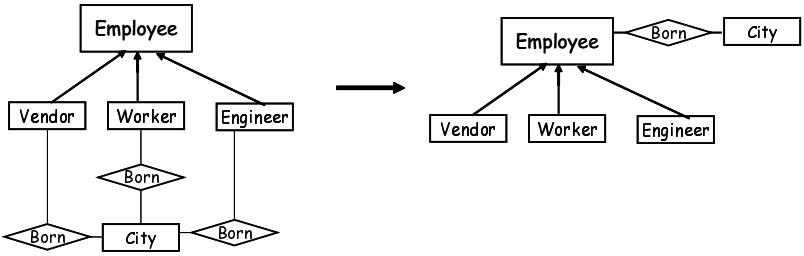



**Fig. 2.18.** An equivalent readable schema

In the relational model, normalization is strictly related to the structure of functional dependencies. Several degrees of normalization have been defined in the relational model, such as first, second, third, Boyce Codd, fourth, and other normal forms. The most popular and intuitive normal form is the *Boyce Codd normal form* (BCNF). A relation schema `R` is in BCNF if for every non trivial functional dependency `X -> Y` defined on `R`, `X` contains a key `K` of `R`, i.e., `X` is a superkey of `R`. For more details on the BCNF and other normal forms, see [11] and [66].

To exemplify, a relational schema `R` is in BCNF if all nontrivial functional dependencies have a key in the left hand side of the dependency, so, all non key attributes depend on a unique key. The interpretation of this property is that the relational schema represents a unique concept, with which all nontrivial functional dependencies are homogeneously associated, and whose properties are represented by all non-key attributes.

As already mentioned, normalization is a property that can be defined in every conceptual or logical model; as an example of normalization not applied to the relational model, Figure 2.19 shows an unnormalized schema in the Entity Relationship model. It is made of a unique entity `Employee-Project`, with five attributes; two of them, the underlined ones, define the identifier of the entity. Following [20], we can define the concept of normalized ER schema

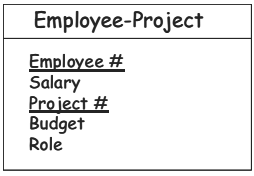



**Fig. 2.19.** An unnormalized Entity Relationship schema

by associating the functional dependencies defined among the attributes of the entity, and adapting the above definition of BCNF to the entities and the relationships. We define the following functional dependencies in the schema:

- `EmployeeId` → `Salary`

- `ProjectId` → `Budget`
- `EmployeeId,ProjectId` → `Role`

that lead to a violation of BCNF. With the objective of normalizing the schema, we can transform the entity `Employee-Project` into a new schema (see Figure 2.20) made of two entities, `Employee` and `Project`, and one many-to-many relationship defined between them. Now the entities and the relationship are in BCNF, as is the whole schema.

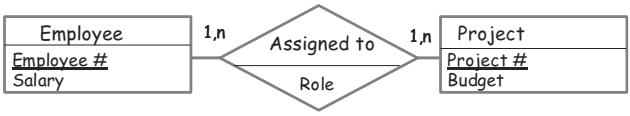



**Fig. 2.20.** A normalized schema

## 2.8 Summary

In this chapter we have seen a variety of dimensions and metrics that characterize the concept of data quality. These dimensions provide a reference framework to those organizations interested in the quality of data, and allow them to characterize and to some extent measure the quality of data sets. Furthermore, fixing and measuring data quality dimensions allow comparison with reference thresholds and values that may be considered target quality values to be achieved in the organization. As a consequence, quality dimensions are at the basis of any process of measurement and improvement of data quality in an organization. As an example, in contracts related to sale of data, the issue of quality of service is crucial, expressing precisely and unambiguously the demand for quality data. Finally, dimensions may be mentioned in laws and rules concerning data usage in government for citizen/business relationships.

It is not surprising that there are many dimensions, since data aim to represent all kinds of spatial, temporal, and social phenomena of the real world; furthermore, in databases, data are represented at two different levels, the intension and the extension, and, consequently, different dimensions have to be conceived. Moreover, we have seen that dimensions may be domain independent, i.e., of general application, or else domain dependent, referring to phenomena characteristic of specific domains. As long as, on one end, ICT technologies evolve, and, on the other end they are applied increasingly to new sciences and applications of the real world, data quality dimensions will evolve and new dimensions will arise. The concept of data is rapidly evolving, from structured data typical of relational databases, to semistructured

data, unstructured data, documents, images, sounds, and maps resulting in a continuous change of the concept of data quality.

Due to the above evolutive phenomena, and the relative immaturity of the data quality research area, another issue that is not surprising, and that is significant in the area, is the absence of enforced de facto standards enacted by international organizations on classifications and definitions of data dimensions and metrics.

Dimensions are the core of any investigation in data quality, and they will be used throughout in the rest of the book.

# 3

# Models for Data Quality

## 3.1 Introduction

In Chapter 2 we introduced several dimensions which are useful to describe and measure data quality in its different aspects and meanings. In order to use database management systems (DBMSs) we represent data, and the relative operations on it, in terms of a *data model* and a *data definition and manipulation language*, i.e., a set of structures and commands that can be represented, interpreted, and executed by a computer. We could follow the same process to represent, besides data, their quality dimensions. This means that in order to represent data quality, we have to extend data models.

Models are widely used in databases for various purposes, such as analyzing a set of requirements and representing it in terms of a conceptual description, called *conceptual schema*; such a description is translated into a *logical schema*; queries and transactions are expressed on such a logical schema.

Models are also used in the wider area of information systems to represent business processes of organizations in terms of subprocesses, their inputs and outputs, causal relationships between them, and functional/non-functional requirements related to processes. Such models are needed in order to help the analyst, e.g., to analyse and foresee process behaviour, measure performance, and design possible improvements.

In this chapter we investigate the principal extensions of traditional models for databases and information systems, proposed to deal with data quality dimensions issues. In Section 3.2 we investigate proposed extensions of conceptual and logical database models for structured data typical of relational DBMSs. Logical models are considered both from the perspective of data description models, and as related to data manipulation and data provenance. Then we discuss models for semistructured data, with specific attention to XML schemas (Section 3.3). In Section 3.4 we move on to management information system models; here, we investigate two "orthogonal" issues: (i) extensions of models for process descriptions to issues related to sources, users

involved in data checks, etc., and (ii) proposals for joint representation of elementary and aggregated data and related qualities. In all the models that we are going to describe, we will see that the extensions of models to data quality issues lead to structures characterized by high complexity.

## 3.2 Extensions of Structured Data Models

The principal database models are the Entity Relationship model, the most common for conceptual database design (see [20]), and the relational model, adopted by a wide range of database management systems.

### 3.2.1 Conceptual Models

Several solutions exist for extending the Entity Relationship model with quality characteristics (see [184] and [183]). The different proposals focus on *attributes*, the unique representation structure in the model with which data values may be associated. A possibility is to model the quality of attribute values as another attribute of the same entity. For example, if we want to express a dimension (e.g., accuracy or completeness) for the attribute `Address` of an entity `Person`, we may add (see Figure 3.1) a new attribute `AddressQualityDimension` to the entity.

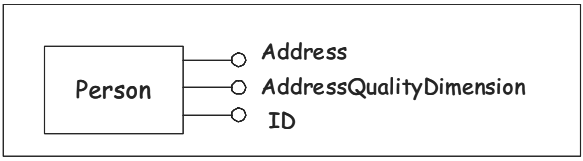



**Fig. 3.1.** A first example of quality dimension represented in the Entity Relationship Model

The drawback of this solution is that now the entity is no longer normalized, since the attribute `AddressQualityDimension` is dependent upon `Address`, which is dependent upon `Id`. Another problem is that if we want to define several dimensions for an attribute, we have to define a new attribute for each dimension, resulting in a proliferation of attributes.

A second possibility is to introduce two types of entities, explicitly defined for expressing quality dimensions and their values: a data quality dimension entity and a data quality measure entity.

The goal of the `DataQualityDimension` entity is to represent all possible pairs of dimensions and corresponding ratings; the pairs <`DimensionName`, `Rating`> constitute the set of dimensions and possible corresponding values resulting from measurements. In the previous definition, we have implicitly assumed that the scale of rating is the same for all attributes. If the

scale depends on the attribute, then we have to extend the properties of the `DataQualityDimension` entity to <`Dimension-Name`, `Attribute`, `Rating`>.

In order to represent metrics for dimensions, and its relationship with entities, attributes, and dimensions, we have to adopt a more complex structure than the one shown in Figure 3.2, in which we introduce the `DataQualityMeasure` entity; its attributes are `Rating`, in which the values depend on the specific dimension modeled, and `DescriptionofRating`. The complete *data quality schema*, which we show by means of the example in Figure 3.2, is made up of

1. The original *data schema*, made in the example of the entity `Class` with all its attributes (here, we represent only the attribute `Attendance`).
2. The `DQ Dimension` entity with a pair of attributes <`DimensionName`, `Rating`>.
3. The relationship between the entity `Class`, the related attribute `Attendance`, and the `DQ Dimension` entity with a many-to-many relationship `ClassAttendanceHas`; a distinct relationship has to be introduced for each attribute of the entity `Class`.
4. The relationship between the previous structure and the `DQ Measure` entity with a new representation structure that extends the Entity Relationship model, and relates entities and relationships.

The overall structure adopted in Figure 3.2 has been proposed in [184].

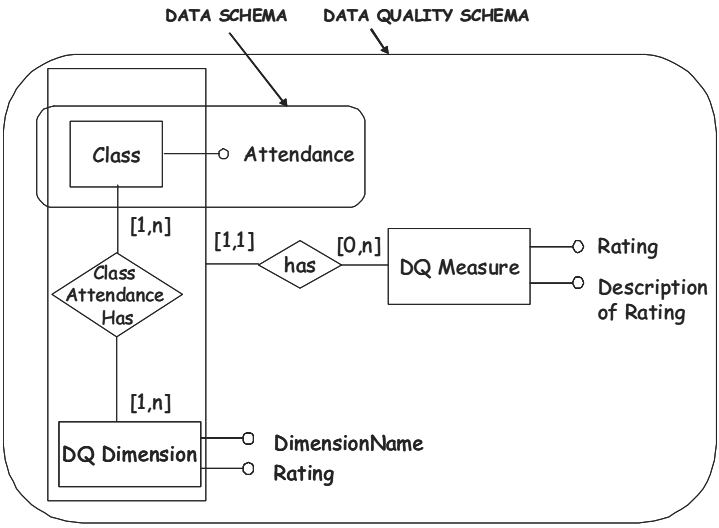



**Fig. 3.2.** A first example of quality dimension represented in the Entity Relationship Model

The above example shows how complex a schema becomes extended with the above structures to describe qualities.

### 3.2.2 Logical Models for Data Description

[204] and [206] extend the relational model with quality values associated with each attribute value, resulting in the *quality attribute model*. We explain the model with an example, shown in Figure 3.3.

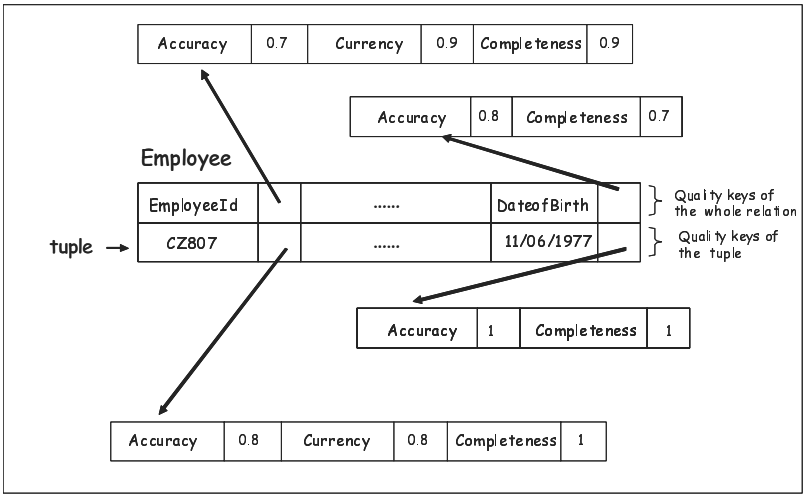



**Fig. 3.3.** An extension of the Relational Model

The figure shows a relational schema `Employee`, defined on attributes `EmployeeId`, `DateofBirth`, and others, and one of its tuples. Relational schemas are extended adding an arbitrary number of underlying levels of *quality indicators* (only one level in the figure) to the attributes of the schema, to which they are linked through a quality key. In the example, the attribute `EmployeeId` is extended with three quality attributes, namely accuracy, currency, and completeness, while the attribute `DateofBirth` is extended with accuracy and completeness, since currency is not meaningful for permanent data such as `DateofBirth`. The values of such quality attributes measure the quality dimensions' values associated with the whole relation instance (top part of the figure). Therefore, completeness equal to 0.7 for the attribute `DateofBirth` means that the 70 % of the tuples have a non-null value for such an attribute. Similar structures are used for the instances level quality indicator relations (bottom part of the figure); if there are `n` attributes of the relational schema, `n` quality tuples will be associated to each tuple in the instance.

### 3.2.3 The Polygen Model for Data Manipulation

In principle, in every process of data collection and analysis, such as medical or biological experiments, data originating from different sources are manipulated in different stages; new data produced at each stage inherit the quality of ancestor data according to histories that depend on the execution plan. In Chapter 4, for several quality dimensions and relational algebra operations, we will investigate the functional relationships between the quality values of the input data and the quality values of the output data. In this section we investigate an extension of the relational model, called *Polygen model* ([202] and [206]), proposed for explicitly tracing the origins of data and the intermediate sources. The model is targeted to heterogeneous distributed systems; the name of the model is derived from "multiple" "sources" (respectively, "poly" and "gen" in Greek). Now we briefly discuss the model, relevant for its pioneer role in the area. A *polygen domain* is a set of ordered triples:

1. a datum drawn from a simple domain in a schema of a local database;
2. a set of *originating databases* denoting the local databases from which the datum originates; and
3. a set of *intermediate databases* in which the data led to the selection of the datum.

A *polygen relation* is a finite set of time varying tuples, each tuple having the same set of attribute values from the corresponding polygen domains. A *polygen algebra* is a set of relational algebra operators whose semantics allows annotation propagation. The five primitive operators in the model are project, cartesian product, restrict, union, and difference. More precisely:

1. *project*, *cartesian product*, *union*, and *difference* are extended from the relational algebra. The difference operator over two Polygen relations $\mathtt{r}_1$ and $\mathtt{r}_2$ is extended as follows (for the remaining operators see [202] and [206]). A tuple $\mathtt{t}$ in $\mathtt{r}_1$ is selected if the data part of $\mathtt{t}$ is not identical to those of the tuples of $\mathtt{r}_2$. Since each tuple in $\mathtt{r}_1$ has to be compared with all the tuples in $\mathtt{r}_2$, it follows that all the originating sources of the data in $\mathtt{r}_1$ are to be included in the intermediate source set produced by the difference operator.
2. The *restrict* operator is introduced to select tuples in a polygen relation that satisfy a given condition, and such tuples populate intermediate sources.
3. *Select* and *join* are defined in terms of the restrict operator, so they also involve intermediate sources.
4. New operators are introduced, e.g. *coalesce*, which takes two columns as input and merges them into one column (no inconsistency is admitted).

Note that in general in heterogeneous multidatabase systems, the values coalesced may be inconsistent. This issue is not considered in the Polygen approach; it will be discussed in detail in Section 6.4.3 dedicated to instance-level conflict resolution techniques.

### 3.2.4 Data Provenance

The Polygen model is a first attempt to represent and analyze the provenance of data, which has been recently investigated in a more general context. *Data provenance* is defined in [36] as the "description of the origins of a piece of data and the process by which it arrived in the database." The typical mechanism to trace the provenance is the use of *annotations* that can be exploited to represent a wide spectrum of information about data, such as comments or other types of metadata, and, in particular, data representing the quality of data. Annotations can be used in a variety of situations including

1. systematically trace the provenance and flow of data, namely even if the data has undergone a complex process of transformation steps, we can determine the origins by examining the annotations;
2. describe information about data that would otherwise have been lost in the database, e.g. an error report about a piece of data;
3. enable the user to interpret the data semantics more accurately, and to resolve potential conflicts among the data retrieved from different sources. This capability is useful in the field of data integration (see Chapter 6), where we are interested in understanding how data in different databases with heterogeneous semantics and different quality levels can be integrated;
4. filter the data retrieved from a database according to quality requirements;
5. improve the management of data trustworthiness through annotations referring to the reputation of a source or to a certification procedure.

Two types of provenance are defined in the literature, *why provenance* and *where provenance* (see [49], [36], and [47] as the main references in this area). We introduce them by means of an example. Assume we issue the following query:

```
SELECT StudentId, LastName, Sex
FROM Student
WHERE Age > SELECT AVERAGE Age FROM Student
```

over the relational schema `Student` (`StudentId`, `LastName`, `Sex`, `Age`).

If the output is the tuple `<03214, Ngambo, Female>`, the provenance of the tuple can be related to two distinct data items:

1. The set of tuples in the input relation that contributed to the final result. In this case, all the tuples have to be selected as contributing tuples, since any modification in one tuple may affect the presence of `<03214, Ngambo, Female>` in the result. This kind of provenance is called *why provenance*, since we are looking for the tuples that explain the shape of the output.

2. The tuple(s) in the input relation that originated the values `03214, Ngambo,` and `Female` in the output tuple. In this case, the set is made up of the unique tuple with `StudentId` = 03214. This kind of provenance is called *where provenance*, since in this case we are interested in finding from where annotations are propagated. In the case of a join between two tuples, both would be considered part of the input set.

The where provenance is particularly useful in the data quality context. In the case where annotations represent quality values, control of the process of quality dimension propagation is allowed by identifying the sources that are responsible for quality degradation. For the above reasons, in the following we focus on the where provenance.

We will discuss the concept of the where provenance and its different meanings in the following context: given a relational database `D`, with a set of annotations associated with tuples in `D`, and a query Q over `D`, compute the provenance of an output tuple `t` in the result of Q.

If we think of possible meanings, i.e., methods to compute the where provenance (similar considerations can be made for the why provenance), two different approaches exist: the *reverse query* (or lazy) approach and the *forward propagation* (or eager) approach.

In the *reverse query approach* (see [49]) and [36]), a "reverse" query Q′ is generated in which the result is the tuple or set of tuples that contribute, when Q has been executed, in producing it.

In the *forward propagation approach*, when applying Q, an enriched query Q* is generated and executed that computes how annotations are propagated in the result of Q. The approach is called *eager*, since provenance is immediately made available, together with the output of Q. The forward propagation approach, in turn, has three possible types of execution or *propagation schemes* [47], called the *default scheme*, the *default-all scheme*, and the *custom propagation scheme*. We introduce the three schemes by means of an example. Assume (see Figure 3.4) we have a database of clients made up of two different tables, `Client1` and `Client2` and a mapping table between identifiers of clients in `Client1` and `Client2`, (a typical situation in many organizations).

Intuitively, the default propagation scheme propagates annotations of data according to where data is copied from. Assume that the following query $Q_1$ is computed on the database of Figure 3.4:

```
SELECT DISTINCT c.Id, c.Description
FROM Client1 c
WHERE c.Id = 071
```

The result of $Q_1$ executed against the relation `Client1` in the default propagation scheme is the unique tuple

$< \texttt{071}[\texttt{ann}_1]; \texttt{Cded}[\texttt{ann}_2] >$

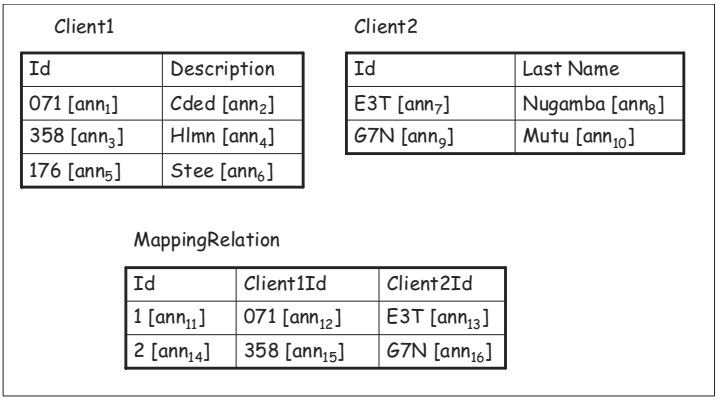



**Fig. 3.4.** Two `Client` relations and a mapping relation

The semantics of the default scheme is quite natural, but it has a drawback, in that two equivalent queries (i.e., queries that return the same output for every database) may not propagate the same annotations to the output. Consider the two queries, $Q_2$:

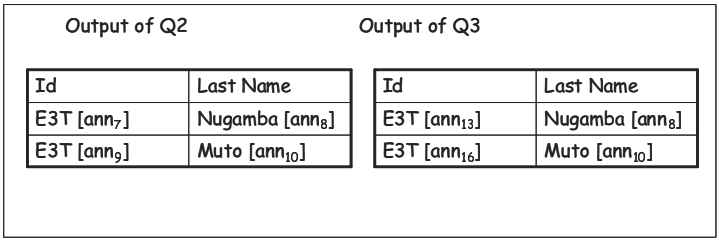
```
SELECT DISTINCT c2.Id AS Id, c2.LastName AS LastName
FROM Client2 c2, MappingRelation m
WHERE c2.Id = m.Client2Id
```

and $Q_3$:

```
SELECT DISTINCT m.Id AS Id, c2.LastName AS LastName
FROM Client2 c2 , MappingRelation m
WHERE c2.Id = m.Client2Id
```

The results of running $Q_2$ and $Q_3$ under the default propagation scheme are shown in Figure 3.5. For $Q_2$ the annotations for the `Id` attribute are from the `Client2` relation while for $Q_3$ the annotations for the `Id` attribute are from the `MappingRelation`.

The *default scheme* propagates the annotation for equivalent queries differently. We need a second propagation scheme, where propagations are invariant under equivalent queries. This scheme is called the *default-all propagation scheme* in [47]; it propagates annotations according to where data is copied from among all equivalent formulations of the given query. In case a user wants to bear the responsibility to specify how annotations should propagate, a third scheme can be adopted, the *custom scheme*, where annotation propagations are explicitly declared in the query.




| Id | Last Name |
|---|---|
| E3T [$ann_7$] | Nugamba [$ann_8$] |
| E3T [$ann_9$] | Muto [$ann_{10}$] |

| Id | Last Name |
|---|---|
| E3T [$ann_{13}$] | Nugamba [$ann_8$] |
| E3T [$ann_{16}$] | Muto [$ann_{10}$] |

**Fig. 3.5.** The output of two queries

The above schemes can be applied flexibly, whatever the type of the annotated information, i.e., it could be the source relation, the exact location within the source, or a comment on the data.

## 3.3 Extensions of Semistructured Data Models

In [175], a model for associating quality values to data-oriented XML documents is proposed. The model, called *Data and Data Quality* ($D^2Q$), is intended to be used in the context of a cooperative information system (CIS). In such systems, the cooperating organizations need to exchange data each other, and it is therefore critical for them to be aware of the quality of such data. $D^2Q$ can be used in order to certify the accuracy, consistency, completeness, and currency of data. The model is semistructured, thus allowing each organization to export the quality of its data with a certain degree of flexibility. More specifically, quality dimension values can be associated with various elements of the data model, ranging from the single data value to the whole data source. The main features of the $D^2Q$ model are summarized as follows:

- A *data class* and a *data schema* are introduced to represent the *domain* data portion of the $D^2Q$ model, namely, the data values that are specific to a given cooperating organization's domain.
- A *quality class* and a *quality schema* correspond to the quality portion of the $D^2Q$ model.
- A *quality association function* that relates nodes of the graph corresponding to the data schema to nodes of the graph corresponding to the quality schema. Quality associations represent biunivocal functions among all nodes of a data schema and all non-leaf nodes of a quality schema.

In Figure 3.6, an example of a $D^2Q$ schema is shown. On the left-hand side of the figure, a data schema is shown representing enterprises and their owners. On the right-hand side, the associated quality schema is represented. Specifically, two quality classes, `Enterprise_Quality` and `Owner_Quality` are associated with the `Enterprise` and `Owner` data classes. Accuracy nodes are shown

for both data classes and related properties. For instance, `Code_accuracy` is an accuracy node associated with the `Code` property, while `Enterprise_accuracy` is an accuracy node associated with the data class `Enterprise`. The arcs connecting the data schema and the quality schema with the `quality labels` represent the quality association functions.

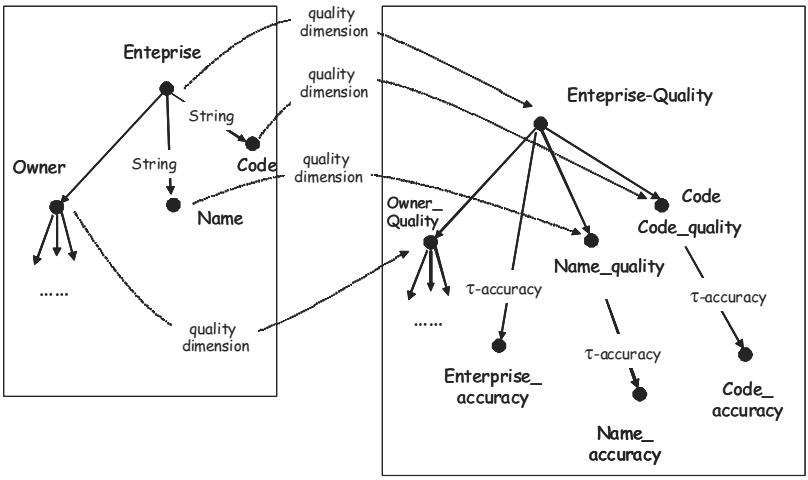



**Fig. 3.6.** Example of $D^2Q$ quality schema

The $D^2Q$ model is intended to be easily translated into the XML data model. This is important for meeting the interoperability requirements that are particularly stringent in cooperative systems. Once translated into XML, the model can be queried by means of an extension of the XQuery language that queries quality values in the model. XQuery allows users to define new functions. Quality values represented according to the $D^2Q$ model can be accessed by a set of XQuery functions, called *quality selectors*. Quality selectors are defined for accuracy, completeness, consistency, currency and for the overall set of quality values that can be associated with a data node.

In Figure 3.7, the implementation of the quality selector *accuracy*() is shown as an example. `Searchroot` is a function defined to reach the root of a document containing the input node.

```
define function accuracy($n as node*) as node* {
let $root := searchroot($n), qualitydoc:=document(string($root/@qualityfile))
for $q in $n/@quality
for $r in $qualitydoc//*[@qOID eq $q]/accuracy
return $r }
```

**Fig. 3.7.** Accuracy selector implementation as an XQuery function

The $D^2Q$ model represents quality values to be associated with generic data. XML is used as a language for modeling quality dimensions in a growing number of contributions. For example, see in [126] a proposal for modeling quality of data by means of six quality measures meaningful in the biological domain. Being domain specific, such a proposal also includes metrics that allow the computation of node quality values across the XML graph, by considering the interdependencies between quality values of the various nodes in the graph.

## 3.4 Management Information System Models

In this section we discuss management information system models in their relation to data quality issues. We discuss process models in Sections 3.4.1 and Section 3.4.2, introducing the IP-MAP model and its extensions. Issues related to data models are discussed in Section 3.4.3.

### 3.4.1 Models for Process Description: the IP-MAP model

The Information Production Map (IP-MAP) model [177] is based on the principle that data can be seen as a particular product of a manufacturing activity, and so descriptive models (and methodologies) for data quality can be based on models conceived in the last two centuries for manufacturing traditional products. The IP-MAP model is centered on the concept of *information product (IP)*, introduced in Chapter 1.

An *information production map* is a graphical model designed to help people comprehend, evaluate, and describe how an information product such as an invoice, customer order, or prescription is assembled in a business process. The IP-MAP is aimed at creating a systematic representation for capturing the details associated with the manufacturing of an information product. IP-MAPs are designed to help analysts to visualize the information production process, identify ownership of process phases, understand information and organizational boundaries, and estimate time and quality metrics associated with the current production process. There are eight types of construct blocks that can be used to form the IP-MAP. Each construct block is identified by a unique name and is further described by a set of attributes (metadata). The content of metadata varies depending on the type of construct block. In Figure 3.8, the possible types of construct blocks are shown, together with the symbol used for their representation.

An example of information production map is shown in Figure 3.9. Information products (IP in the figure) are produced by means of processing activities and data quality checks on *raw data* (RD), and semi-processed information or *component data* (CD), introduced in Chapter 2. In the example, we assume that high schools and universities of a district have decided to cooperate in order to improve their course offering to students, avoiding

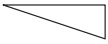
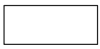
| Concept name | Symbol | Description |
|---|---|---|
| Source (raw input data) |  | Represents the source of each raw (input) data that must be available in order to produce the information product expeceted by the customer |
| Customer (output) |  | Represents the consumer of the information product. The consumer specifies the data elements that constitute the "finished" information products. |
| Data quality |  | Represents the checks for data quality on those data items that are essential in producing a "defect-free" information product. |
| Processing |  | Represents any calculations involving some or all of the raw input data items or component data items required to ultimately produce the information block. |
| Data Storage |  | It is any data item in a database. |
| Decision |  | It used to describe the different decision conditions to be avaluated and the corresponding procedures for handling the incoming data items, based on the evaluation. |
| Business Boundary |  | Specifies the movement of the information product accross departmental or organization boundaries. |
| Information system boundary |  | Reflects the changes to the raw data items or component data items as they move form one information system to another type of information system. These system changes could be inter or intra business units. |

**Fig. 3.8.** IP-MAP construct blocks

overlappings and being more effective in the education value chain. To this end, high schools and universities have to share historical data on students and their curricula. Therefore, they perform a record linkage activity that matches students in their education life cycle. To reach this objective, high schools periodically supply relevant information on students; in case it is in paper format, the information has to be converted in electronic format. At this point invalid data are filtered and matched with the database of university students. Unmatched students are sent back to high schools for clerical checks, and matched students are analyzed; the result of the analysis on curricula and course topics are sent to the advisory panel of the universities.

### 3.4.2 Extensions of IP-MAP

The IP-MAP model has been extended in several directions. First, more powerful mechanisms have been provided in [160] and [174], called *event process chain diagrams* representing the *business process overview*, the *interaction model* (how company units interact), the *organization model* (who does what), the *component model* (what happens), and the *data model* (what data is needed). This is done by modeling

- the event that triggers the use of data by a process;
- the communication structure between sources, consumers, and organizational groups;

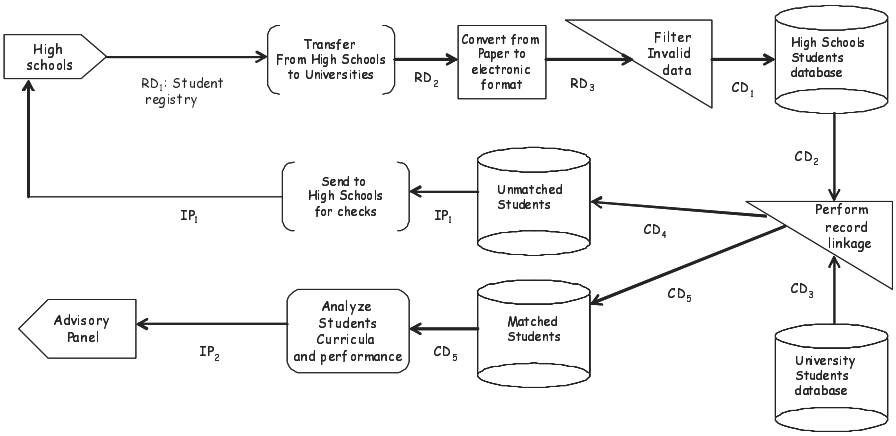



**Fig. 3.9.** An example of IPMAP

- the hierarchy of organizational groups/functions;
- the relationship between products, storages, and other data components;
- logical relationships between events and processes.

A modeling formalism is proposed in [174], called IP-UML, extending UML with a data quality profile based on IP-MAP. The use of UML instead of the IP-MAP formalism has the following advantages:

1. UML is a standard language, and computer-aided tools have been implemented for it;
2. UML is a language supportive of analysis, design, and implementation artifacts, so the same language can be used in all the phases of analysis and development;
3. the expressive power of UML is higher with reference to the process modelling constructs.

We briefly recall that in UML (see [150], and [79]) the specification of analysis and design elements is based on the notion of a *model element*, defined as an abstraction drawn from the system being modeled; the principal model elements are *classes* and *relationships* between classes. A *constraint* is a semantic restriction that can be attached to a model element. A *tag definition* specifies new kinds of properties that may be attached to model elements. A *tagged value* specifies the actual values of tags of individual model elements. A *stereotype* is a new model element that extends previously defined model elements through a precise semantics. According to the UML specification [148] "a coherent set of such extensions, defined for a specific purpose, constitutes a *UML profile*."
The starting concepts of IP-UML are the ones defined in the IP-MAP framework; the result of the proposed extension is a UML profile called data quality

profile. The *data quality profile* consists of three different models, namely, the data analysis model, the quality analysis model and the quality design model.

The *data analysis model* specifies which data are important for consumers, as its quality is critical for the organization's success. In the data analysis model information products, raw data and component data are represented as a stereotyped UML class. A *quality data class* is a class labeled with this a that generalizes Information product classes, Raw data classes, and Component data classes.

The *quality analysis model* consists of modeling elements that can represent quality requirements of data, related to one of the dimensions typically defined for data quality. The set of dimensions proposed consists of four categories; for example the *intrinsic information quality category* includes accuracy, objectivity, believability, and reputation. In order to model the overall set of dimension-related requirements, the following stereotypes are introduced:

1. A *quality requirement* class generalizes the set of quality requirements that can be specified on a quality data class.
2. A *quality association* class associates a quality requirement class with a quality data class. Quality requirements on data need to be verified so that, if they are not satisfied, improvement actions can be taken; therefore, a constraint is specifically introduced on the quality association.

The specification of a distinct stereotype for each quality requirement has the advantage of clearly fixing the types of requirements that can be associated with data.

The *quality design model* specifies IP-MAPs. The IP-MAP dynamic perspective, in which processes are described together with exchanged data, can be obtained by combining *UML activity diagrams* with *UML object flow diagrams*. Activity diagrams are a special case of state diagrams in which the states are action or subactivity states and in which the transitions are triggered by completion of the actions or subactivities in the source states. Object flows are diagrams in which objects that are input or output from an action may be shown as object symbols. The following UML extensions need to be introduced, to represent IP-MAP elements:

- *stereotyped activities*, to represent processing and data quality blocks;
- *stereotyped actor*, to represent customer, source, and data storage blocks;
- *stereotyped dependency relationships*, to give a precise semantics to the relationships between some elements.

Notwithstanding the rich set of new structures introduced in the extensions of IP-MAP, such extensions suffer from different limitations, discussed in the next section, with new models that attempt to override such limitations.

### 3.4.3 Data Models

A first limitation of IP-MAP (and IP-MAP extensions) lies in the fact that it does not distinguish between or provide specific formalisms for *operational*

*processes*, which make use of *elementary data*, and *decisional processes*, which use *aggregated data*. The information system of an organization is composed of both types of data, that present different quality problems. So, it seems relevant to enrich data models for management information systems to explicitly provide a uniform formalism to represent both types of data and their quality dimensions.

Secondly, IP-MAP does not take specific features of cooperative information systems (CIS) into account. In a CIS, as Figure 3.10 shows, an organization can be modeled as a collection of processes that transform input information flows into output information flows, and that carry a stream of information products. In Figure 3.10, three organizations are represented that exchange four information flows: two of them are composed of two information products each; the two remaining flows exchange one single information product. In the domain of a specific organization, an input flow to a process can be transformed into (i) an internal flow, (ii) an input to another intra-organizational process, or (iii) an output flow to one or more external organizations.

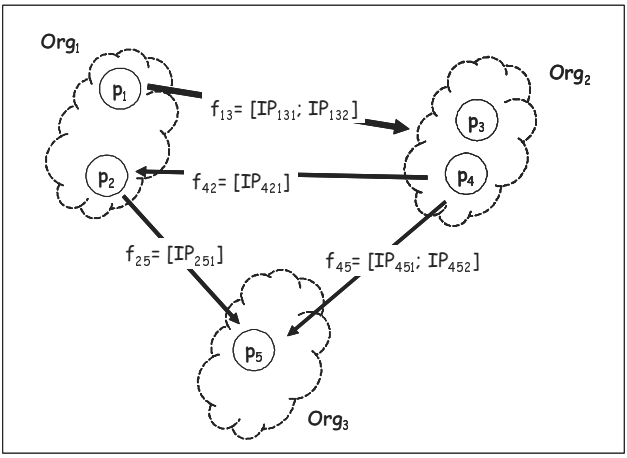



**Fig. 3.10.** Organizations, processes, and information flows in a Cooperative Information System

In [131], [130], and [132], a comprehensive approach to overcome the above limitations is presented, discussed in the following sections.

### A Data Model of the Information Flows of an Organization

We first distinguish two different roles for organizations exchanging information flows in a CIS, namely, a *producer (organization)* when it produces flows for other organizations, and a *consumer (organization)* when it receives flows

from other organizations. Every organization usually plays both roles. Following traditional manufacturing practice, we characterize the quality of the individual items produced on the producer side; by extension, we associate a *quality offer profile* to a producer organization. Such a profile represents the quality that the organization is willing to offer to its customers, i.e., to other consumer organizations that require that information for use in a cooperative process. Symmetrically, on the consumer side we define the notion of *quality demand profile* to express acceptable quality levels for the information items that consumers will acquire. Ultimately, we frame the problem of managing information quality within an organization as the problem of matching the quality profile offered by that organization to the quality requested by the consumers of the organization. At this point, we are able to define a framework for expressing quality offer and demand in a CIS context. The framework models both the structure of a cooperative organization (*data schema*) and its quality profiles (*quality schema*, see next section) in a uniform, hierarchical way.

We start by associating quality profiles with the elementary information items that the organization produces and consumes during the execution of processes (see Figure 3.11 for the metaschema of the data schema, represented with a class diagram in UML).

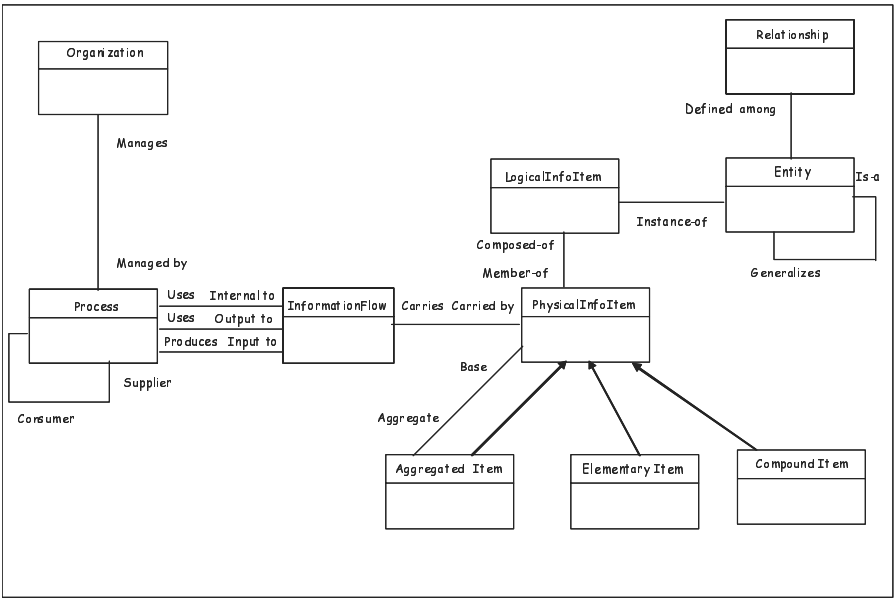



**Fig. 3.11.** Data, process, and organization schema

An *information flow* f is a sequence of *physical information items* (PII), that are streamed from a producer process to one or more consumer pro-

cesses. For instance, given a domain entity `Address`, and its instance `4 Heroes Street` (suitably identified using keys defined for `Address`), a PII would be a specific copy of J. Smith's address, produced at a particular time t by a process $p_1$ and sent to a process $p_2$ over flow f. All PIIs produced by any process at any time, referring to the same data, homogeneous in meaning, are associated with a single *logical information item* `4 Heroes Street`.

Physical information items and logical information items describe *atomic* (or *elementary*) information items and their flow in time. As the metaschema in Figure 3.11 shows, a *compound item* is obtained recursively from other compound or elementary items using composition functions, such as the record type function (e.g. an `Address` is composed of `Street`, `City`, and `ZipCode`). An *aggregated item* is obtained from a collection of elementary and compound items by applying an aggregation function to them (e.g., the average income of tax payers in a given town).

With the above representation structures we are able to model both information flows made of elementary items and flows made of aggregated items. Finally, we associate information flows between processes, and processes with organizations. Information flows are of three types: input to, output from, and internal to processes. We enrich the set of representation structures with other structures, typical of a conceptual model, such as *entity*, *relationship* among entities, and *generalization* among entities, as done in the schema in Figure 3.11, with usual meanings in the Entity Relationship model.

### A Quality Profile Model

In order to represent and compute quality profiles, associated with all the classes in the previous schema, we model the quality profile of an organization as a *data cube* on a given set of dimensions, using the *multidimensional database model* proposed in [3]. We view the quality profile of a single item as one point in a *multidimensional cube*, in which the axes include a hierarchy of entities consisting of physical and logical information items, flows, processes, organizations, and quality dimensions.

The information carried by each quality point in the resulting *quality cube* is the single quality measurement at the finest level of granularity, i.e., the quality descriptor associated with a single physical data item and for a single dimension. Figure 3.12 shows the *star schema*, in the data warehouse approach; it has the quality values as fact entity, and the remaining ones as the dimension entities; attributes of fact and dimension entities are not shown.

The quality profiles for information flows, processes, and entire organizations are computed as appropriate aggregations from a base quality cube. Thus, once an appropriate set of *aggregation functions* (e.g., average) is defined over quality descriptors, quality profiles at each level of granularity in an organization are described within an established framework for multidimensional data. As an example, consider again Figure 3.10, where two organizations, five processes and four flows are defined. We may aggregate quality

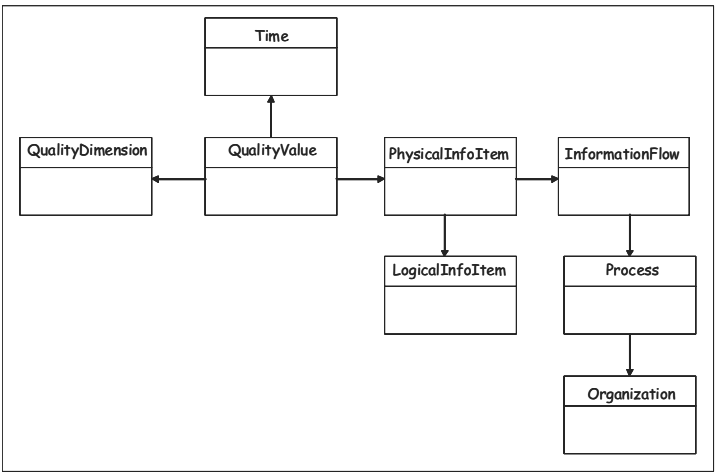



**Fig. 3.12.** Star schema of the data quality cube

values along the following chain: (i) physical information item, (ii) information flow, (iii) process, (iv) organization; and, using aggregation functions, we may associate quality values with each one of the above information flows, processes, and organizations, according to the perspective we choose.

## 3.5 Summary

In this chapter we have seen several proposals for extending data and process models, to provide them with structures for representing quality dimensions and for using them to measure and improve data quality profiles of single information flows, processes, and entire organizations. In the following chapters we will address the core topics of research in and experience with data quality, i.e., techniques and methodologies proposed for DQ measurement and improvement. We anticipate that such techniques and methodologies seldom rely on the proposals presented in this chapter on model extensions, with the distinctive exception of the IP-MAP model. Furthermore, only a few prototypical DBMSs have experienced the adoption of some of the approaches mentioned, among them [6]. This feeble connection is due to the complexity of the overall equipment of the representational structures proposed in the different approaches, and the lack of consolidated tools and DBMSs to manage them.

The future of research on models appears to be in provenance and trustworthiness issues. In cooperative information systems, and peer-to-peer information systems, knowing the provenance and the trustworthiness of data is crucial for the user, who may trace the history of data and increase his or her awareness in accessing and using them.

# 4

# Activities and Techniques for Data Quality: Generalities

In Chapter 1 we noticed that data quality is a multifaceted concept, and the cleaning of poor quality data can be performed by measuring different dimensions and setting out on several different activities, with different goals. A *data quality activity* is any process we perform directly on data to improve their quality. An example of "manual" data quality activity is the process we perform when we send an e-mail message, and the e-mail bounces back because of an unknown user; we check the exact address in a reliable source, and we type the address on the keyboard more carefully to avoid further mistakes. An example of "computerized" data quality activity is the matching of two files in which inaccurate records are included, in order to find similar records that correspond to the same real-world entity through an approximate method. Other activities for improving data quality act on processes; they will be discussed and compared with data quality activities in Chapter 7.

Data quality activities are performed using different techniques that result in different efficiency and effectiveness for measuring and improving data quality dimensions. The final goal of this chapter, and of Chapters 5 and 6, is to define the data quality activities and introduce the most relevant techniques proposed to support each of them. In this chapter we first define the activities (Section 4.1) and provide the reader a map of the book sections where the different activities are dealt with. The two most investigated data quality activities, namely object identification and data integration, will be discussed in Chapter 5 and Chapter 6. In this chapter we discuss two of the activities, namely, quality composition (Section 4.2), and error localization and correction (Section 4.3). The final section (Section 4.4) opens the discussion on costs and benefits of data quality, introducing and comparing proposed classifications for costs/benefits; this material will be applied in Chapter 7 on methodologies.

## 4.1 Data Quality Activities

Although a large quantity of algorithms, heuristics, and knowledge-based techniques have been proposed that are classified as *data quality activities*, a limited number of categories can be identified. They are listed in the following, providing temporary definitions that will be detailed later in this chapter, as well as in Chapters 5 and 6:

1. *New data acquisition* is a process of data acquisition performed with the goal of refreshing the database with new quality data. The manual example discussed above falls in this category.
2. *Standardization* (or *normalization*) is the modification of data with new data according to defined standard or reference formats, e.g., change of Bob to Robert, change of Channel Str. to Channel Street.
3. *Object identification* (or *record linkage*, *record matching*, *entity resolution*), given one or more tables, has the purpose of identifying those records in the tables that represent the same real-world object. When the table is unique this activity is also called *deduplication*.
4. *Data integration* is the task of presenting a unified view of data owned by heterogeneous and distributed data sources. Data integration has different goals resulting in two specific activities:
   - *quality-driven query processing* is the task of providing query results on the basis of a quality characterization of data at sources;
   - *instance-level conflict resolution* is the task of identifying and solving conflicts of values referring to the same real-world objects.
5. *Source trustworthiness* has the goal of rating sources on the basis of the quality of data they provide to other sources in an open or peer-to-peer context, where no or little control exists on the quality of data.
6. *Quality composition* defines an algebra for composing data quality dimension values, for instance, given two relations in which the completeness values are known, and an operator, e.g., the union, computes the completeness of the union, starting from the completeness of the operand relations.
7. *Error localization* (or *error detection*), that given one or more tables, and a set of semantic rules specified on them, finds tuples that do not respect such rules.
8. *Error correction*, that, given one or more tables, a set of semantic rules, and a set of identified errors in tuples, corrects erroneous values in tuples in order to respect the overall set of rules.
9. *Cost optimization* has the goal to optimize a given target on data quality, according to a cost objective. For example, among different providers of data sets characterized by different costs and quality dimension metrics, we could be interested in selecting the provider with the optimal cost/quality ratio for a given data demand.

Other activities that more loosely pertain to data quality are

- *Schema matching*, which takes two schemas as input and produces a mapping between semantically correspondent elements of the two schemas.
- *Schema cleaning*, which provides rules for transforming the conceptual schema in order to achieve or optimize a given set of qualities (e.g., readability, normalization) while preserving other properties (e.g., equivalence of content).
- *Profiling* analyzes data in the database in order to infer intensional properties, such as the structure of the database, fields with similar values, join paths, and join sizes.

Since schema matching, schema cleaning, and schema profiling primarily involve data schemas, they will not be considered in the following. Two of the activities, namely, object identification/record linkage and data integration are of crucial importance in current business scenarios, and have been widely investigated from a research and industrial perspective. As already mentioned, two specific chapters are dedicated to them; Chapter 5 will describe object identification and Chapter 6 will describe data integration. In addition,

1. New data acquisition will be dealt with in Chapter 7 in the context of data quality improvement methodologies, where it will be discussed as one of the data driven strategies.
2. Standardization is usually performed as a preprocessing activity in error localization, object identification, and data integration. However, as standardization is mostly included in object identification techniques, we describe in detail in Chapter 5 as one of the steps of object identification.
3. Source trustworthiness is an emerging research issue in open and peer-to-peer systems. When dealing with such systems, trust and data quality become two crucial concepts. We will discuss such issues in Chapter 9, dedicated to open research problems.
4. Cost optimization covers four different aspects: (i) cost trade-offs between quality dimensions, discussed in Chapter 2; (ii) cost and benefit classifications for characterizing data quality in business processes, addressed in Section 4.4.1; (iii) cost/benefit analysis of data quality improvement processes, described in Chapter 7; and (iv) cost-based selection of data sources, illustrated in Chapter 9, in which the cost of data will be one of the parameters that guide the decision process.

In the rest of this chapter, we briefly describe the remaining activities. The following sections deal with quality composition (Section 2), error localization and correction (Section 3), and, finally with cost and benefit classifications (Section 4).

## 4.2 Quality Composition

In several contexts, including e-Business and e-Government, especially when data is replicated across different sources, it is usual to obtain new data by

combining data sets extracted from one or more sources. In these contexts, it is important to be able to calculate a quality dimension or the set of qualities of the new resulting data, starting from the quality dimension values of the original sources, if available. Furthermore, in order to enhance the quality of data, it is often not enough to consider single sources and independently orchestrate improvement actions on them; instead, such actions should be properly complemented by composing data from different sources.

Let us consider a set of public administrations that cooperate with each other in an e-Government scenario, and let us focus on a specific data quality dimension, namely, the completeness dimension. In some countries, in every municipality the following registries are held: (i) a personal data registry for residents and (ii) a separate registry for the civil status of the residents. At the regional level, we may assume that there are local income tax payer registries, while at central level there are usually national social insurance, accident insurance and other registries. These sources usually have different levels of completeness in representing the corresponding reality of interest, and in many administrative processes, these sources are combined. It would be interesting to directly calculate the completeness of the combined result starting from the completeness of the sources, if known, without performing on the result a costly process of quality measurement. This is the goal of the data quality composition activity.

The general problem statement for the definition of the quality composition problem is represented in Figure 4.1. The data source, or the set of data sources, X, described according to a data model M, is processed by a generic composition function F. It is defined on a set of operators O = $[o_1, \ldots, o_k]$ defined in the model M. Also, a function $Q_D$ calculates the value of the quality dimension D for X, i.e., $Q_D$(X) and the value of D for Y equals F(X), i.e. $Q_D$(Y). We aim to define the function $Q_D^F$(X) that calculates $Q_D$(Y) starting from $Q_D$(X), instead of calculating such a value directly on Y by applying the function $Q_D$(Y).

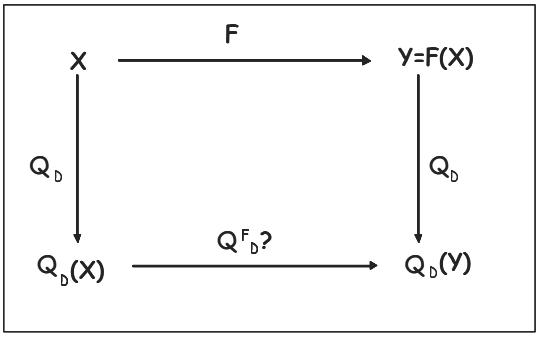



**Fig. 4.1.** The general problem of quality composition

We will consider the case of this problem in which

- M is the relational model;
- O corresponds to the set of relational algebraic operators, such as Union, Intersection, Cartesian product, Projection, Selection, and Join;
- D is a specific data quality dimension, e.g. completeness or accuracy; and
- $Q_D^F$ is a function that evaluates the quality of the relations under different hypotheses and for different relational operators.

The problem of defining a *composition algebra* for data quality dimensions has been considered in several papers in the literature, namely, Motro and Ragov [136]; Wang et al. [206]; Parsiann et al. [157], [155], [156]; Naumann et al. [140], and Scannapieco and Batini [173]. In Figure 4.2 these approaches are compared on the basis of (i) the adopted model, (ii) the quality dimensions considered, (iii) the relational algebra operators taken into account, and (iv) the specific assumptions on the sources. In the following section we comment all the issues dealt with in Figure 4.2; when describing the approaches, we will use the names of the authors in the first column of the table.

| Paper | Model | Specific assumptions on the sources | Quality dimensions considered | Algebraic operators |
|---|---|---|---|---|
| Motro 1998 | Relational model with OWA (implicit) | No assumption | Soundness<br>Completeness | Cartesian Product<br>Selection<br>Projection |
| Parssiann 2002 | Relational model with OWA (implicit) | Uniformely distributed errors in identifier attributes<br>Error probabilites for all attributes independent of each other<br>Uniformely distributed errors in non identifier attributes for mismember and other tuples | Accuracy<br>Inaccuracy<br>Mismembership<br>Incompletness | Selection<br>Projection<br>Cartesian Product<br>Join |
| Wang 2001 | Relational model | Uniformly distributed errors | Accuracy | Selection<br>Projection |
| Naumann 2004 | Data integration system<br>Set of data sources + Universal relation with CWA | Set relationships between sources<br>- Disjointness<br>- Quantified overlap<br>- Independence (coincidential overlap)<br>- Containment | Coverage<br>Density<br>Completeness | Join merge<br>Full outer join merge<br>Left outer join merge<br>Right outer join merge |
| Scannapieco 2004 | Relational model with OWA and CWA | Open world vs closed world assumption<br>Set relationships between sources<br>- Disjointness<br>- Nonquantified overlap<br>- Containment | Completeness | Union<br>Intersection<br>Cartesian product |

**Fig. 4.2.** Comparison between approaches to quality composition

We recall that in Chapter 2 we have introduced the concepts of *closed world assumption*, *open world assumption*, *reference relation*, and the related

dimension definitions of *value completeness*, *tuple completeness*, and *relation completeness*.

### 4.2.1 Models and Assumptions

Motro and Parssian consider a model in which an ideal (called conceptual by Parssian) relation *r-ideal* and the corresponding real relation *r-real* can be constructively defined; as a consequence, they may distinguish common and non common tuples between them. Motro defines dimensions in terms of the differences between r-ideal and r-real, measured considering, respectively, common tuples and uncommon ones. Parssian goes further, distinguishing, between the two types of tuples, between pairs of tuples that differ in the primary keys (called *identifiers* in the Parssian approach and in the following), and tuples that are identical on the keys and differ on the non key attributes (*non-identifier* attributes in the following). The assumptions dealt with by Parssian on error probabilities both on identifier and non identifier attributes are described in Figure 4.2. Wang is not interested in completeness issues. He does not consider tuples that are in the ideal relation and are not members of the real relation; furthermore, he assumes that the tuples that appear in the real relation are only there by mistake, called *mismember tuples*. Wang, within his simplified model, assumes uniform distribution of errors in the relation.

Naumann, differently from other authors, investigates quality composition in the context of a data integration system. Naumann adopts a model where data sources correspond to local relations and databases. A global source exists, called *universal relation*, that corresponds to the set of all tuples that can be obtained through the sources at hand. Naumann considers four different cases of set relationships between sources: (i) disjointness, (i) containment, (iii) independence, corresponding to coincidental overlap, and (iv) quantified overlap, where the number of common tuples among sources is known. In the following, we will describe the set of operators adopted by Naumann, both in expressing the relationship between the sources and the universal relation, and in the characterization of quality composition. Naumann is interested in evaluating the quality of the process of composing sources, in order to put together information that is split into different sources. For this reason, he is interested in evaluating the behavior of join operators.

The *full outer join merge* operator is defined as a suitable adaptation of the full outer join operator of relational algebra (see [66]) to the context in which conflicts in tuples are taken into account. In the proposed model, it is assumed that tuples of different sources have been identified as corresponding to the same object of the real world. When we merge two tuples $\mathtt{t}_1$ and $\mathtt{t}_2$ referring to the same object, depending on the situations common attributes can have (i) both null values, (ii) $\mathtt{t}_1$ a null value and $\mathtt{t}_2$ a specified value, (iii) the inverse, i.e., $\mathtt{t}_1$ a specified value and $\mathtt{t}_2$ a null value, (iv) the same specified value, and (v) different specified values. In the last case, it is assumed that

a *resolution function* is provided. Let us consider two given sources, corresponding to relations $\mathtt{r}_1$ and $\mathtt{r}_2$ The *join merge* operator may be defined as an extension of the join operator by further applying the resolution function. The *full* (and the *left/right*) *outer join merge* operator(s) are defined as an extension of the outer join operators, where join merge is used instead of join. The *universal relation* is defined as the full outer join merge of $\mathtt{r}_1$ and $\mathtt{r}_2$. Within this model, Naumann adopts the closed world assumption, since only tuples in the sources may exist in the universal relation.

Scannapieco adopts both closed world and open world assumptions; in this way, all the types of completeness discussed in Chapter 2 may be defined. Furthermore, in the open world assumption, given two distinct relations $\mathtt{r}_1$ and $\mathtt{r}_2$, two different hypotheses can be made on the reference relations: (i) the two reference relations of $\mathtt{r}_1$ and $\mathtt{r}_2$ are the same, and (ii) the reference relations differ. This is due to the fact that, when composing relations with composition operators such as union or join, we may give (see Figure 4.3) two different interpretations to the operations, according to the following assumptions:

- if the two reference relations are the same (left-hand side of Figure 4.3), incompleteness concerns the lack of objects with sources referring to the same reality of interest; and
- if the two reference relations are different (right-hand side of Figure 4.3), the interpretation of the composition results in the integration of different realities of interest.

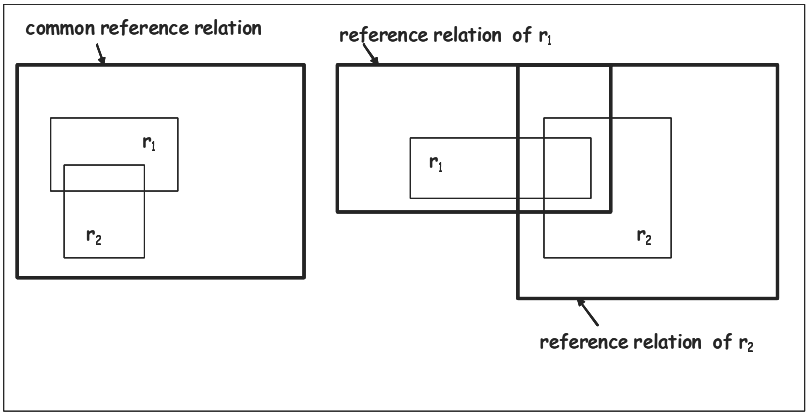



**Fig. 4.3.** Assumptions for reference relations

In the two previous cases, the evaluation of the resulting completeness has to be different. With reference to set relationships between sources, Scannapieco considers overlap, containment, and a weaker notion of overlap, where the number of common tuples is not known.

### 4.2.2 Dimensions

In this section we first discuss dimensions comparatively, then we focus on two specific dimensions, namely accuracy and completeness.

In Motro, given an ideal relation `r-ideal` and the corresponding real relation `r-real`, two dimensions are defined:

- *Soundness* measures the proportion of the real data that is true:

$$\frac{|\texttt{r-ideal}| \cap |\texttt{r-real}|}{|\texttt{r-real}|}$$

- *Completeness* measures the proportion of the true data that is stored in the real relation:

$$\frac{|\texttt{r-ideal}| \cap |\texttt{r-real}|}{|\texttt{r-ideal}|}$$

Parssian defines four different dimensions, depending on the pair of tuples considered in the relationship between the ideal relation and the real relation. More precisely

- A tuple in `r-real` is *accurate* if all of its attribute values are accurate, i.e., are identical to the values of a corresponding tuple of `r-ideal`. We call $S_{accurate}$ the set of accurate tuples.
- A tuple is *inaccurate* if it has one or more inaccurate (or null) values for its non-identifier attributes, and no inaccurate values for its identifier attribute (or attributes); $S_{inaccurate}$ is the set of inaccurate tuples.
- A tuple is a *mismember* if it should not have been captured into `r-real`, but has been; $S_{mismember}$ is the set of mismember tuples.
- A tuple belongs to the *incomplete* set $S_{incomplete}$ if it should have been captured into `r-real`, but has not been.

In Figure 4.4 we show an example of (i) an ideal relation `Professor`; (ii) a possible corresponding real relation, with accurate tuples in white, inaccurate tuples in pale gray, and mismember tuples in dark gray; and (iii) a set of incomplete tuples. Accuracy, inaccuracy, mismembership of `r-real` are defined, respectively, as

$$accuracy = \frac{|S_{accurate}|}{|\texttt{r-real}|},$$

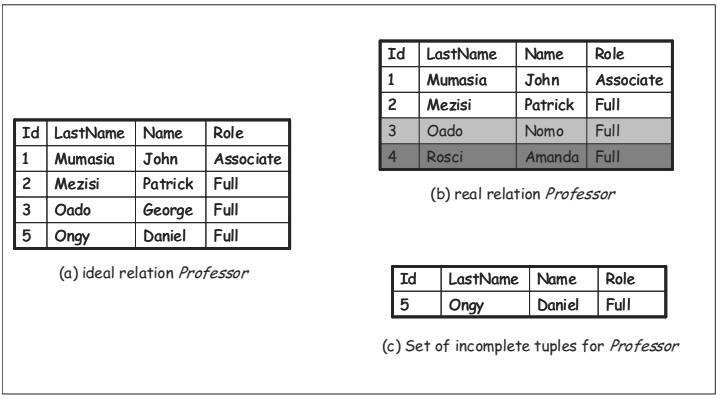
$$inaccuracy = \frac{|S_{inaccurate}|}{|\texttt{r-real}|},$$

$$mismembership = \frac{|S_{mismember}|}{|\texttt{r-real}|}.$$

The completeness of `r-real` can be defined as

$$\frac{|S_{incomplete}|}{|\texttt{r-real}| - |S_{accurate}| + |S_{incomplete}|}$$

since, when considering `r-real`, we have to eliminate mismember tuples and add the set of incomplete tuples.



| Id | LastName | Name | Role |
|---|---|---|---|
| 1 | Mumasia | John | Associate |
| 2 | Mezisi | Patrick | Full |
| 3 | Oado | George | Full |
| 5 | Ongy | Daniel | Full |

(a) ideal relation *Professor*

| Id | LastName | Name | Role |
|---|---|---|---|
| 1 | Mumasia | John | Associate |
| 2 | Mezisi | Patrick | Full |
| 3 | Oado | Nomo | Full |
| 4 | Rosci | Amanda | Full |

(b) real relation *Professor*

| Id | LastName | Name | Role |
|---|---|---|---|
| 5 | Ongy | Daniel | Full |

(c) Set of incomplete tuples for *Professor*

**Fig. 4.4.** Examples of accuracy/inaccuracy/mismember tuples and incomplete set in the Parssian approach

Wang, within the concept of accuracy, distinguishes between a *relation accuracy* and a *tuple accuracy*. In the hypothesis of uniform distribution of errors that cause inaccuracy, the tuple accuracy is defined as *probabilistic tuple accuracy*. It coincides numerically with the overall relation accuracy.

In Naumann, completeness is analyzed from three different points of views, corresponding to the coverage, density, and completeness dimensions.

1. The *coverage* of a source `s` captures the number of objects represented in the source `s` with respect to the total number of objects in the universal relation `ur`, and is defined as

$$\frac{|\,\texttt{s}\,|}{|\,\texttt{ur}\,|}.$$

2. The *density* of a source captures the number of *values* represented in the source, and is defined as the number of non-null values referred to by the attributes in the universal relation. More formally, we first define the *density of an attribute* `a` of `s` as

$$d(\texttt{a}) = \frac{|\,(\texttt{t} \in \texttt{s} \mid \texttt{t.a} \neq null)\,|}{|\,\texttt{s}\,|}.$$

The *density* of the source s is the average density over the set of all attributes A of the universal relation ur:

$$\frac{1}{|\,\mathtt{A}\,|}\sum_{\mathtt{a}\in\mathtt{A}} d(\mathtt{a}).$$

3. The *completeness* of a source s captures the number of values represented in the source, with respect to the total potential amount of values of the real world; it is expressed by the formula

$$\frac{|\,(\mathtt{a}_{ij} \neq null \mid \mathtt{a}_{ij} \in \mathtt{s})\,|}{|\,\mathtt{ur}\,| \times |\,\mathtt{A}\,|},$$

where $\mathtt{a}_{ij}$ is the value of the $j_{th}$ attribute of tuple $\mathtt{t}_i$ in s.

Scannapieco considers all the dimensions presented for completeness in Chapter 2, and also other ones (the interested reader can refer to [173]).

In the rest of the section, we provide various results on accuracy and completeness. Due to previously discussed heterogeneity of approaches, we will discuss each proposal separately. Due to the more significant contributions provided, in the following we focus on Wang, Parssian, Naumann, and Scannapieco. We adopt the symbols described in Figure 4.5.

| Symbol | Meaning |
|---|---|
| r | input relation |
| $r_1, r_2, .., r_n$ | a set of n input relations |
| s | output relation |
| \| r \| | size of the relation r |
| acc | accuracy |
| inacc | inaccuracy |
| cov | coverage |
| compl | completeness |

**Fig. 4.5.** Symbols used in the exposition

### 4.2.3 Accuracy

Wang provides several results for selection and projection operators. We analyze selection, while for the more complex formulas related to projection, we refer you to [206]. Under the assumption that | s |, the size of the output relation, is available, the following formula easily derives from the hypothesis of uniform distribution of errors:

$$\mathtt{acc(s)} = \mathtt{acc(r)}.$$

Other formulas are provided for the worst and best case scenarios; for instance, for the worst case, if $|\mathtt{r}| \leq |\mathtt{s}|$, then $\mathtt{acc(s)} = 0$. See [206] for more details.

Results provided by Parssian are richer, due to the the larger set of dimensions defined for the input relations. We provide details for accuracy and inaccuracy in the case of cartesian product and selection operations.

For cartesian product, applied to two relations $\mathtt{r}_1$ and $\mathtt{r}_2$, the following formulas can be simply derived:

$$\mathtt{acc(s)} = \mathtt{acc(r}_1) * \mathtt{acc(r}_2)$$

and

$$\mathtt{inacc(s)} = \mathtt{acc(r}_1) * \mathtt{inacc(r}_2) + \mathtt{acc(r}_2) * \mathtt{inacc(r}_1) + \mathtt{inacc(r}_1) * \mathtt{inacc(r}_2).$$

Concerning the selection operation, four different cases apply according to the structure of the condition in the selection: the selection condition applies to an identifier/non-identifier attribute and the selection is an equality/inequality. We will examine two of them.

In the case where the condition is an inequality applied to an identifier attribute, due to the assumption of uniform distribution of errors, the accuracy, inaccuracy, mismembership, and completeness values for $\mathtt{s}$ are identical to the ones for $\mathtt{r}$. This is because the status of the selected tuples remains unchanged.

In the case where the condition is an equality applied to a non-identifier attribute $\mathtt{A}$, tuples are selected or not selected depending on their being accurate or inaccurate in the values of $\mathtt{A}$. To estimate the size of the various components of $\mathtt{s}$, we need to estimate the probability that an accurate/inaccurate/mismember tuple is in one of the parts of $\mathtt{r}$ related to non identifier attributes appearing or not appearing in the condition. We call $P(t \in s)$ such probability. The formula for accuracy in this case is intuitively:

$$\mathtt{acc(s)} = \mathtt{acc(r)} * \frac{|\mathtt{r}|}{|\mathtt{s}|} * P(\mathtt{t} \in \mathtt{s})$$

For a formal proof of the previous formula and details on all remaining cases, see [156].

### 4.2.4 Completeness

In the following we focus on the contributions by Naumann and Scannapieco. First, in the Naumann approach there is a functional relationship between completeness, coverage, and density of a relation $\mathtt{r}_1$, namely,

$$\mathtt{compl(r}_1) = \mathtt{cov(r}_1) * \mathtt{density(r}_1).$$

This relationship results directly from the definitions provided. Naumann characterizes the composition functions, in the case of binary operators on

two relations $r_1$ and $r_2$, for the three dimensions and all the previously defined operators under the assumptions defined in Section 4.2.1.

In Figure 4.6 we show several cases for the coverage dimension, which we discuss here; for other cases, see [140].

| Assumption/ operator | $r_1$ and $r_2$ disjoint | Quantified overlapping (= x) | $r_1$ contained in $r_2$ |
|---|---|---|---|
| Join merge | 0 | \|x\| / \|ur\| | cov($r_1$ ) |
| Left outer join merge | cov($r_1$ ) | cov(r1) | cov($r_1$ ) |
| Full outer join merge | cov($r_1$ ) + cov($r_2$ ) | cov($r_1$ ) + cov($r_2$ ) - \|x\| / \|ur\| | cov($r_1$ ) |

**Fig. 4.6.** Coverage composition functions in Naumann

Looking at Figure 4.6, in the case of the join merge, the results of the operator under the different assumptions are, respectively, (i) no object, (ii) only the common objects, and (iii) only the objects of $r_1$, leading straightforwardly to the formulas. In the case of the left outer join merge, due to the property of the left outer join of maintaining all the tuples of the first source $r_1$ in the result, the coverage is independent of the assumptions, and is equal to cov($r_1$). Similar considerations hold for the full outer join merge case. For all the other cases and properties not mentioned here, we refer to [140].

In the approach of Scannapieco, we consider the two cases of the open world assumption, in which given $r_1$ and $r_2$ input relations are defined, respectively, over (i) the same reference relation, or (ii) two different reference relations. Note that we assume to know the sizes of the reference relations themselves, and not the reference relations themselves. We consider the evaluation of completeness for the union operator.

*Case 1: Same Reference Relation.* We suppose that

$$\texttt{ref}(r_1) = \texttt{ref}(r_2) = \texttt{ref}(s).$$

In the case in which no additional knowledge on relations is available, we may only express an upper bound:

$$\texttt{compl}(r) \geq \max\ (\texttt{compl}(r_1), \texttt{compl}(r_2)).$$

Behind this inequality, we can distinguish three more cases:

1. disjointness: if $r_1 \cap r_2 = 0$ then $\texttt{compl}(s) = \texttt{compl}(r_1) + \texttt{compl}(r_2)$;
2. non quantified partial overlap: if $r_1 \cap r_2 \neq 0$ then $\texttt{compl}(s) > \max(\texttt{compl}(r_1), \texttt{compl}(r_2))$; and

| Id | LastName | Name | Role |
|---|---|---|---|
| 1 | Ongy | Daniel | Full |
| 2 | Mezisi | Patrick | Full |
| 3 | Oado | George | Full |
| 4 | Rosci | Amanda | Full |

(a) dept1

| Id | LastName | Name | Role |
|---|---|---|---|
| 1 | Mumasia | John | Associate |
| 2 | Mezisi | Patrick | Full |
| 3 | Oado | George | Full |
| 4 | Gidoy | Nomo | Associate |
| 5 | Rosci | Amanda | Full |

(b) dept2

| Id | LastName | Name | Role |
|---|---|---|---|
| 1 | Mumasia | John | Associate |
| 2 | Oymo | Vusi | Associate |
| 3 | Msgula | Luyo | Associate |
| 4 | Keyse | Frial | Associate |

(c) dept3

| Id | LastName | Name | Role |
|---|---|---|---|
| 1 | Ongy | Daniel | Full |
| 2 | Oado | George | Full |

(d) dept4

**Fig. 4.7.** Examples of input relations

3. containment: if $\mathtt{r}_1 \subset \mathtt{r}_2$ then $\mathtt{compl}(\mathtt{s}) = \mathtt{compl}(\mathtt{r}_2)$.

For example, Figures 4.7a and 4.7b show the two relations `dept1` and `dept2`, each representing professors of a department and having the same reference relation, `ref-dept` = ref(`dept1`) = ref(`dept2`), corresponding to all the professors of the department. Notice that `dept1` represents only full professors. We have the following input data: (i) $|\mathtt{dept1}| = 4$, (ii) $|\mathtt{dept2}| = 5$, and (iii) $|\mathtt{ref\text{-}dept}| = 8$. Hence, $\mathtt{compl}(\mathtt{dept1}) = 0.5$ and $\mathtt{compl}(\mathtt{dept2}) = 0.625$. From this information we can derive

$$\mathtt{compl}(\mathtt{dept1} \cup \mathtt{dept2}) \geq 0.625.$$

Figure 4.7c shows the relation `dept3`, the size of which is 4; this relation contains only associate professors; therefore, $\mathtt{dept3} \cap \mathtt{dept1}$ is Ø. In this case, we can easily compute

$$\mathtt{compl}(\mathtt{dept1} \cup \mathtt{dept3}) = 0.5 + 0.5 = 1$$

Figure 4.7d shows the relation `dept4`, the size of which is 2; observe that $\mathtt{dept4} \subseteq \mathtt{dept1}$. In this case, we have

$$\mathtt{compl}(\mathtt{dept1} \cup \mathtt{dept4}) = 0.5.$$

*Case 2: Different Reference Relations.* We consider a case that can occur in real scenarios, i.e., the reference relations are a disjoint and complete partition of a domain. This is the case, for example, when we merge two disjoint sets of citizens resident in differen cities. More specifically, we suppose that $\mathtt{ref}(\mathtt{r}_1) \cap \mathtt{ref}(\mathtt{r}_2) = \varnothing$ and $\mathtt{ref}(\mathtt{s}) = \mathtt{ref}(\mathtt{r}_1) \cup \mathtt{ref}(\mathtt{r}_2)$. In this case, it is easy to show that the completeness of s for the union is

$$\mathtt{compl}(\mathtt{s}) = \frac{|\mathtt{r}_1| + |\mathtt{r}_2|}{|\mathtt{ref}(\mathtt{r}_1)| + |\mathtt{ref}(\mathtt{r}_2)|} =$$

$$= \frac{\texttt{compl}(\texttt{r}_1) * |\texttt{r}_1| + \texttt{compl}(\texttt{r}_2) * |\texttt{r}_2|}{|\texttt{ref}(\texttt{r}_1)| + |\texttt{ref}(\texttt{r}_2)|}.$$

For other cases, related to intersection and cartesian product, we refer you to [173].

## 4.3 Error Localization and Correction

In the introduction of this chapter, error localization and error correction activities were identified as data quality activities. Error localization and correction are useful every time data have been collected from error-prone sources (e.g., those in which manual input has been performed) or acquired from sources whose reliability is not known at all.

In Chapter 2 we have seen that errors in data may be expressed in terms of a wide number of dimensions; for some of them we have provided measures and, in the case of consistency, formal models to characterize the dimension. We argue that corresponding methods for error localization and correction depend on the type of quality dimension we want to control and achieve. The following sections take into account such dimension dependence, and are hence organized as follows

1. localize and correct inconsistencies in Section 4.3.1;
2. localize and correct incomplete data in Section 4.3.2;
3. localize outliers, i.e., data values that are anomalous with respect to other data, and usually are an indicator of incorrect data, in Section 4.3.3.

### 4.3.1 Localize and Correct Inconsistencies

Historically, the problem of localizing inconsistencies has occurred in statistical surveys carried out by processing answers obtained through a collection of questionnaires, and is also typical of data collected in experiments and analyses (e.g., clinical) for medical diagnosis and care. Error localization and correction is becoming increasingly important when using sensor networks, e.g., for detection of harmful biological and chemical agents and in collecting data in monitoring environmental conditions. The error rate of these sensor networks is highly dependent on the current battery level of the device, interference, and other parameters.

A first formalization of the problem appears in [76]; more recent contributions appear in several papers (see [33], [215], and [163]). In the following, we will consider data collected through questionnaires as a reference case; as we will see, the approach can be generalized to other cases where more complex data models are defined, e.g., relational data model with integrity constraints.

When designing a questionnaire, the data provided as responses to the questionnaire must verify a set of properties, corresponding to the edits introduced in Chapter 2. In the statistical world, the set of all edits is called the

*set of edit rules*, or *check plan*, or *compatibility plan*. Usually, such rules are known only to a certain extent, since collecting and expressing rules is a costly activity, and even a simple questionnaire can result in tens and hundreds of such rules. Errors, or inconsistencies between answers or out-of-range answers, can be due to low quality in the original design of the questionnaire, or can be introduced during any later phase of data production, such as data input or conversion.

When edits are collected, it is crucial that they be proven to be *consistent*, i.e. without contradictions, otherwise, every conceivable procedure to use edits in order to localize errors will fail. Furthermore, they should be *non-redundant*, i.e. no edit in the set can be logically derived from other edits.

As an example of an inconsistent set of edits, assume a survey is performed on the employees of a company. Consider the three edits (here, and in the following, we informally introduce the syntax and the semantics of edits):

1. `Salary = false`, which means "every employee has a salary."
2. `Has a desk = false`, which means "every employee has a desk."
3. `(Salary = true)` and `(Has a desk = true)`, which means "an employee is not allowed to have a salary and to have a desk."

There is an evident contradiction among the three edits. This is an indication that one of the edits, most probably edit 3, is wrong. An example of a redundant set of edits is:

1. `Role = professor` $\wedge$ `AnnualIncome` $<$ `100.000`
2. `AnnualIncome` $<$ `100.000`

where the redundancy concerns the constraint on `AnnualIncome`.

Once we have a *valid*, i.e., at least consistent, set of edits, we can use them to perform the activity of *error localization*. This may be done by checking if the truth assignments associated with the values in the questionnaire satisfy the logic formula corresponding to the set of edits. In this activity, it would be obviously preferable to have a *non redundant* set of edits, because decreasing the number of edits while maintaining the same power of inconsistency detection can simplify the whole process.

After the localization of erroneous records, in order to correct errors, we could perform on them the activity called *new data acquisition* in Section 4.1. Unfortunately, this kind of activity is usually very costly, and, in all the contexts in which data are collected for statistical purposes, the use of edits is usually preferred to correct erroneous data. The activity of using edits to correct erroneous fields by restoring correct values is called *error correction* or *imputation*. The problem of localizing errors by means of edits and imputing erroneous fields is usually referred to as the *edit-imputation problem*. Fellegi and Holt in [76] provide a theoretical model for the edit-imputation problem. The main goals of the model are as follows

- The data in each record should satisfy all edits by changing the fewest fields possible. This is called the *minimum change principle*.

- When imputation is necessary, it is desirable to maintain the marginal and joint frequency distribution of values in the different fields.

The above two goals may be in conflict, as the following example shows. Consider a questionnaire that collects several properties of people, such as <Age, MaritalStatus, TypeofWork>. A "true" record such as <68, married, retired> could result due to some error into <6, married, retired>. Such a record does not respect an edit such as

Age $<$ 15 $\wedge$ MaritalStatus = married.

We may correct 6 into 15, respecting the minimum change principle for the age, but if we apply the rule in all similar cases we alter the distribution of values of Age. Even changing 6 (and analogous incorrect values) in order to respect the frequency distribution of correct values of Age, we could modify the joint distribution with MaritalStatus and TypeofWork. Thus, in general, we have to perform more complex and wide changes. Fellegi and Holt provide a solution to the edit imputation problem that finds the minimum number of fields to change in order to respect all the edits, thus achieving the first goal. They make an important assumption in their method: that implicit edit is known. *Implicit edits* are those that can be logically derived from explicitly defined edits. In *error localization* they were considered redundant edits, and so they were minimized; during *error correction* they cannot be ignored, since they express properties that do not fail for a record but may fail as values are changed. The following example adapted from [215] provides intuition for computational issues. Consider a record,

<Age, MaritalStatus, Relationship-to-Head-of-Household>,

and the following two edits:

edit1: Age $<$ 15 $\wedge$ MaritalStatus = married
edit2: MaritalStatus = not married $\wedge$
Relationship-to-Head-of-Household = spouse

An implicit edit, as may easily be checked, is

edit3: Age $<$ 15 $\wedge$ Relationship-to-Head-of-Household = spouse

We initially assume that edit3 is hidden. Consider now a record $\mathtt{r}_1$ = <10, not married, spouse>. The record fails for edit2; in order to correct the record, we may change the marital status to married, to obtain a new record $\mathtt{r}_2$ that now fails for edit1. So, we have to make a second attempt, that involves the value spouse. If we explicitly consider edit3, we immediately

reach the conclusion that at least one of the two values <10,spouse> has to be changed.

Assuming availability of implicit edits, Fellegi and Holt formulate the problem as a set covering problem. Alternatively, if implicit edits are not available, then the edit-imputation problem can be solved by integer programming methods which are much slower. Probabilistic imputation methods have to be used to deal with the second goal, namely, to maintain the marginal and joint frequency distribution of variables. We refer to [33] for these issues.

### 4.3.2 Incomplete Data

In Chapter 2 we introduced completeness as a relevant data quality dimension, and we defined and provided metrics for it in the context of relational tables. Another type of incompleteness arises in the measurement of phenomena during a period of time, e.g., in time series. We consider now the two cases of completeness.

With regard to relational tables, enforcing explicit values for an attribute $\mathtt{A}$, or for a set of attributes $\mathtt{A}_1$, $\mathtt{A}_2$,..., $\mathtt{A}_n$ in place of missing ones, can be expressed as the problem of conformance to edits of the form

$\mathtt{A}_1$ = null or $\mathtt{A}_2$ = null or ...or $\mathtt{A}_n$ = null.

In this case, the problem of finding the minimum number of values to be modified is trivial, since this number coincides with the set of missing values. Thus, the goal that becomes critical is to maintain the marginal and joint frequency distributions of the attributes. If the attributes to be considered are $\mathtt{A}_1$, $\mathtt{A}_2$,..., $\mathtt{A}_n$, an assumption can be made that attributes are missing monotonically, that is, $\mathtt{A}_i$ is not missing only if $\mathtt{A}_{i-1}$, $\mathtt{A}_{i-2}$, ..., $\mathtt{A}_1$ are not missing. In this case, a regression method can be performed recursively, generating valid values from $\mathtt{A}_1$ to $\mathtt{A}_n$.

With regard to time series, two types of incompleteness can be identified, namely, truncated data and censored data. *Truncated data* corresponds to observations that are dropped from the analyzed data set. For example, customers that take at the most one flight a year might not be included in an airline customer database. *Censored data* correspond to data that we know for sure have not been collected before a certain time $t_1$ (*left censored data*) or after a certain time $t_2$ (*right censored data*). As an example of left censored data, assume we are interested in measuring the *mean time between failure* of a computer; we could have only historical data available after a certain time $t_1$, and we might not know at what time $t_0 < t_1$ the computer started operating. The possible situations are shown in Figure 4.8.

Note that truncated or censored data can also appear in relational tables with values not time stamped. For instance, a 64-bit integer cannot represent values higher than $2^{64} - 1$; so, integer overflows correspond to censored values. As another example, a sales invoice system may assign a default date for

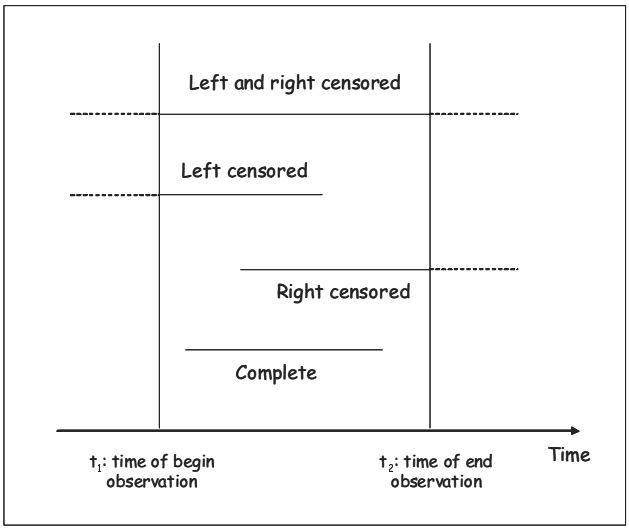



**Fig. 4.8.** Types of incomplete data in time series

missing date invoices. As a consequence, invoices with missing values all have exactly the same data, which has a high frequency.

Truncated and censored data can be detected with the help of histograms and frequency distributions. For example in the sales order system, corresponding to the default date a spike appears in the frequency distribution of dates.

### 4.3.3 Discovering Outliers

A value that is unusually larger or smaller in relation to other values in a set of data is called an *outlier*. As an example, consider the following data:

$$2, 5, 6, 3, 8, 76, 4, 3, 7.$$

Intuition tells us that 76 is a suspicious value, because all the other data are numbers between 0 and 10. Typically, an outlier is attributable to one of the following causes in the measurement of data:

1. it is incorrectly observed, recorded, or entered in a database;
2. it comes from a different population, in relation to other values; and
3. it is correct, but represents a rare event.

In our example, 76 could be a simple typo, where the separating comma between 7 and 6 is missing. This is an example of temporary false or spurious value, sometimes called *data glitch*, that corresponds to causes 1 and 2. It is important to distinguish between outliers of type 3, correct but rare data, and

outliers of types 1 and 2, i.e., data glitches. As a consequence of the above discussion, methods for managing outliers are characterized by two phases, (i) discovering outliers and (ii) deciding between rare data and data glitches.

Outliers are detected by measuring the departure of values from what we expect them to be. We discuss the following methods that can be used for the detection of outliers: control charts, distributional outliers, and time series outliers. A comprehensive list of these methods is discussed in detail in [50].

- *Control charts* have been developed primarily by the manufacturing industry to measure the quality of products; several data samples are collected, and statistics, such as mean and standard error, are computed and analyzed. As an example, in Figure 4.9, the region inside the rectangle represents values that are inside single attribute error limits, while the ellipse represents the joint control limits based on the joint distribution of the two attributes. Some points that are inside control limits of the single attributes are outliers when the elliptic control area corresponding to the pair of attributes is considered.

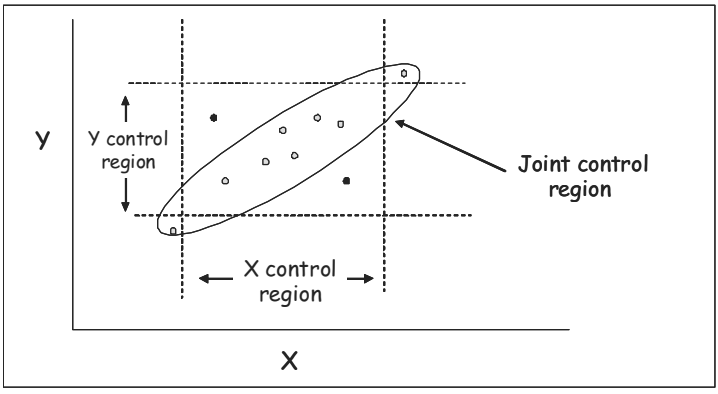



**Fig. 4.9.** Example of a control chart based on two attributes

Control charts are suitable for studying one or two attributes at a time. They cannot be used for capturing outliers based on interrelationships between attributes; it is possible that a value might be well suited in relation to any given attribute, but might be outside fixed error bounds in relation to the attributes taken together.

- *Distributional outliers.* According to this method, outliers are seen as points which are in a region of low density. Since these points are relatively isolated, they are "probably" outliers. The intuition is that outliers are likely to be at a large distance from the other data points. Starting from this intuition, distributional outliers can be found computing the value $F[d](x)$ for every point $x$ in the set of values, which is the fraction of points in the set of values at distance $d$ or more from $x$. The set of

$F[p, d]$ outliers is the set of points $x$ such that $F[d](x) > p$, where $p$ is a threshold value. Note that outliers could be clustered, e.g., because of default or censored values for some of the fields. The threshold $p$ should be adjusted to take these fields into account.

- *Time series outliers.* These methods analyze outliers in time series. They consider relevant properties of time series, such as the fact that data which are close in time tend to be highly correlated. They also consider the presence of cyclic patterns in the data, such as credit card payments that may have peaks at certain hours in the week. A technique for time series starts with partitioning the group of attributes measured in series (such as, e.g., `<CreditCardNumber, Expense>`) into sections, using a space partitioning strategy. Each class of the partition is a state that a data point can have in time. A given time series is modeled as a trajectory of states, with transition probabilities between states. Thus transitions can be ranked by their likelihoods, and outliers correspond to low likelihood transitions.

Once the outliers are identified, we have to decide whether they represent an abnormal but legitimate behavior or a data glitch. In the time series methods, two different measures of deviation are considered for the decision. The *relative deviation* represents the movement of a data point relative to other data points over time. For instance the data points may represent the history of credit card purchases of a customer, with some customers purchasing at a faster rate, while other customers continue at the same rate at which they started. The *within deviation* measures the dynamics of a data point in relation to its own expected behavior.

We briefly compare the two strategies. The relative deviation is more robust, since state changes require significant changes in attributes. The within deviation is sensitive to minor changes and is better for analyzing long-term changes; thus, it is more suitable for discriminating between rare data and glitches. In fact, genuine changes are usually persistent over time, whereas glitches appear and disappear unpredictably. A drop in revenues at a single point in time is more likely to be a data problem, such as missing data, rather than a downward trend. Patterns in glitches reveal systematic causes, such as data in particular missing intervals.

## 4.4 Cost and Benefit Classifications

In this section we start to discuss how an organization can analyze whether it is convenient or not to engage DQ improvement campaigns. In other words, we will discuss how quantifying (i) the costs of current poor data quality, (ii) the costs of DQ initiatives to improve it, and (iii) the benefits that are gained from such initiatives. Cost-benefit analysis is an arduous task in many cost domains, and it is more arduous in the DQ area due to the less consolidated nature of the discipline. The existing proposals range from classifications provided for

costs and benefits to methodologies for performing the cost-benefit analysis process. Classifications are either generic, or specific, e.g., for the financial domain. The advantages of generic classifications (see also [70]) range from establishing clearer terminology to providing consistent measurement metrics. They can be used as checklists during the cost-benefit analysis activity. We discuss in this section issues related to generic classifications, and postpone to Chapter 7 the discussion on methodologies. In the following we distinguish the differences between cost issues and benefit issues.

### 4.4.1 Cost Classifications

Three very detailed classifications for costs appear in English [68], Loshin [123], and Eppler and Helfert [70]. We first present the three classifications, discussing their original issues; then, we propose a common classification framework to compare them all.

The English classification is shown in Figure 4.10. Data quality costs correspond to costs of business processes and data management processes due to poor data quality. Costs for information quality assessment or inspection measure data quality dimensions to verify that processes are performing properly. Finally, process improvement and defect prevention costs involve activities to improve the quality of data, with the goal of eliminating, or reducing, the costs of poor data quality. Costs due to low data quality are analyzed in depth in the English approach, shown in the Figure 4.10, and are subdivided into three categories:

1. *Process failure costs* result when poor quality information causes a process not to perform properly. As an example, inaccurate mailing addresses cause correspondence to be misdelivered.
2. *Information scrap and rework*. When information is of poor quality, it requires several types of defect management activities, such as reworking, cleaning, or rejecting. Examples of this category are
   - redundant data handling, if the poor quality of a source makes it useless, time and money has to be spent to collect and maintain data in another database;
   - business rework costs, due to re-performing failed processes, such as resending correspondence, as in the previous example;
   - data verification costs, when data users do not trust the data, they have to perform their own quality inspection, to remove low quality data.
3. *Loss and missed opportunity costs* correspond to the revenues and profits not realized because of poor information quality. For example due to low accuracy of customer e-mail addresses, a percentage of customers already acquired cannot be reached in periodic advertising campaigns, resulting in lower revenues, roughly proportional to the decrease of accuracy in addresses.

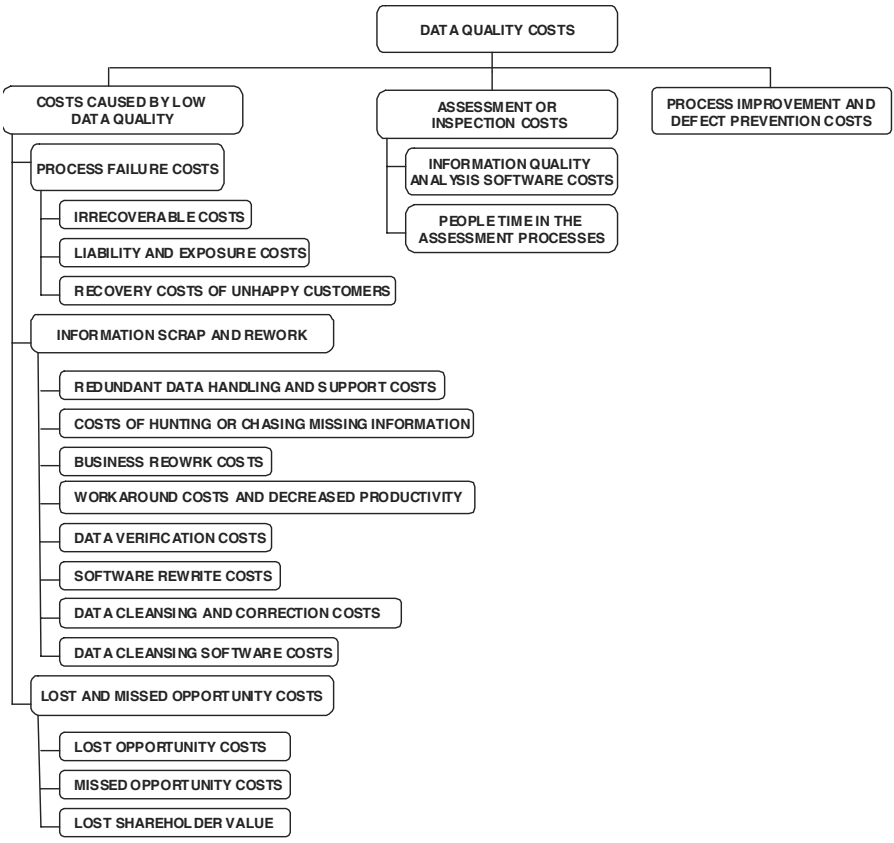



**Fig. 4.10.** The English classification

The Loshin classification is shown in Figure 4.11. Loshin analyzes the costs of low data quality, classifying it in different domain impacts, on

- the operational domain, which includes the components of the system used for processing information and the costs of maintaining the operation of the system;
- the tactical domain, which attempts to address and solve problems before they arise;
- the strategic domain, which stresses the decisions affecting the longer term.

For both the operational impact and tactical/strategic impact several cost categories are introduced. Here, we describe some of the operational impact costs:

- detection costs are incurred when a data quality problem provokes a system error or processing failure;
- correction costs are associated with the actual correction of a problem;

- rollback costs are incurred when work that has been performed needs to be undone;
- rework costs are incurred when a processing stage must be repeated;
- prevention costs arise when a new activity is implemented to take the necessary actions to prevent operational failure due to a detected data quality problem.

Examples of tactical/strategic costs are: (i) delay, due to inaccesible data resulting in a delayed decision process that, in turn, may cause productivity delays, (ii) lost opportunities, i.e., the negative impact on potential opportunities in strategic initiatives, and (iii) organizational mistrust, due to the decision of managers, unsatisfied by inconsistencies in data, to implement their own decision support system, resulting in redundancies and inconsistencies due to frequent use of the same sources.

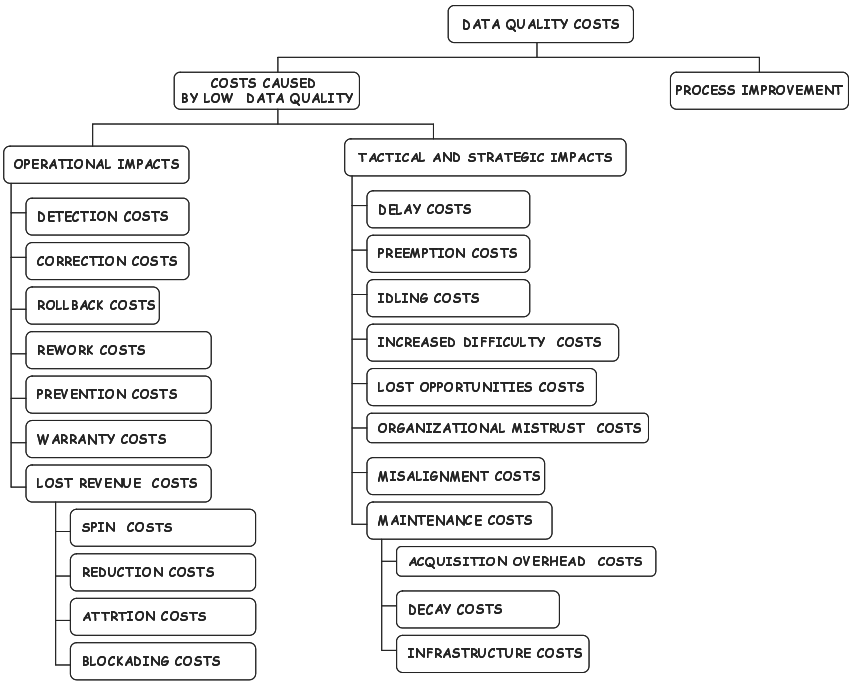



**Fig. 4.11.** The Loshin classification

The EpplerHelfert classification is shown in Figure 4.12. EpplerHelfert derives its classification with a bottom up approach; first, it produces a list of specific costs that have been mentioned in the literature, such as higher maintenance costs and data re-input costs. Then, it generates a list of direct costs associated with improving or assuring data quality, such as training

costs of improving data quality know-how. At this point it puts together the two classifications corresponding to the two major classes of costs, namely cost due to poor data quality and improvement costs. Costs due to poor data quality are categorized in terms of their measurability or impact, resulting in direct vs. indirect cost classes. *Direct costs* are those monetary effects that arise immediately from low data quality, while *indirect costs* arise from the intermediate effects. Improvement costs are categorized within the information quality process.

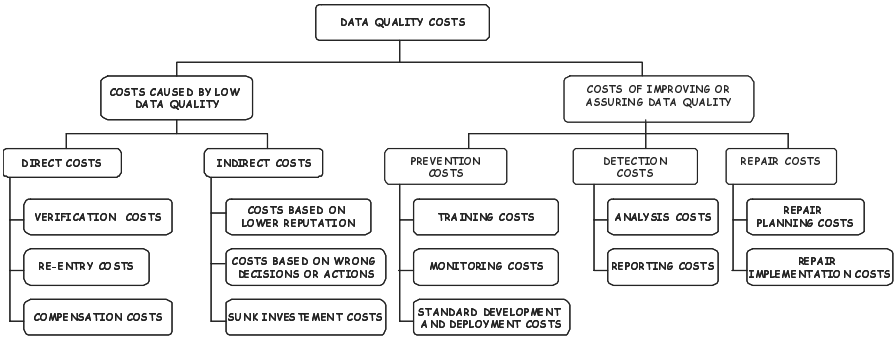



**Fig. 4.12.** The EpplerHelfert classification

For the purpose of producing a new classification that allows for the integration of the three classifications discussed above, we use a second classification proposed by Eppler and Helfert in [70]; such a classification produces a conceptual framework that could be used in the cost-benefit analysis of data quality programs. It is based on the data production life cycle approach, which distinguishes between *data entry*, *data processing*, and *data usage* costs. The iterative attribution of all the cost categories of the three previous classifications to this new high-level classification leads to the comparative classification of Figure 4.13; the different background patterns used for the English, Loshin, and EpplerHelfert classification items are shown in the legend. When comparing the three classifications, we notice that they have very few items in common, all placed at an abstract level, namely *corrective* costs, *preventive* costs, and *process improvement* costs and the two most similar classifications are the English and Loshin ones.

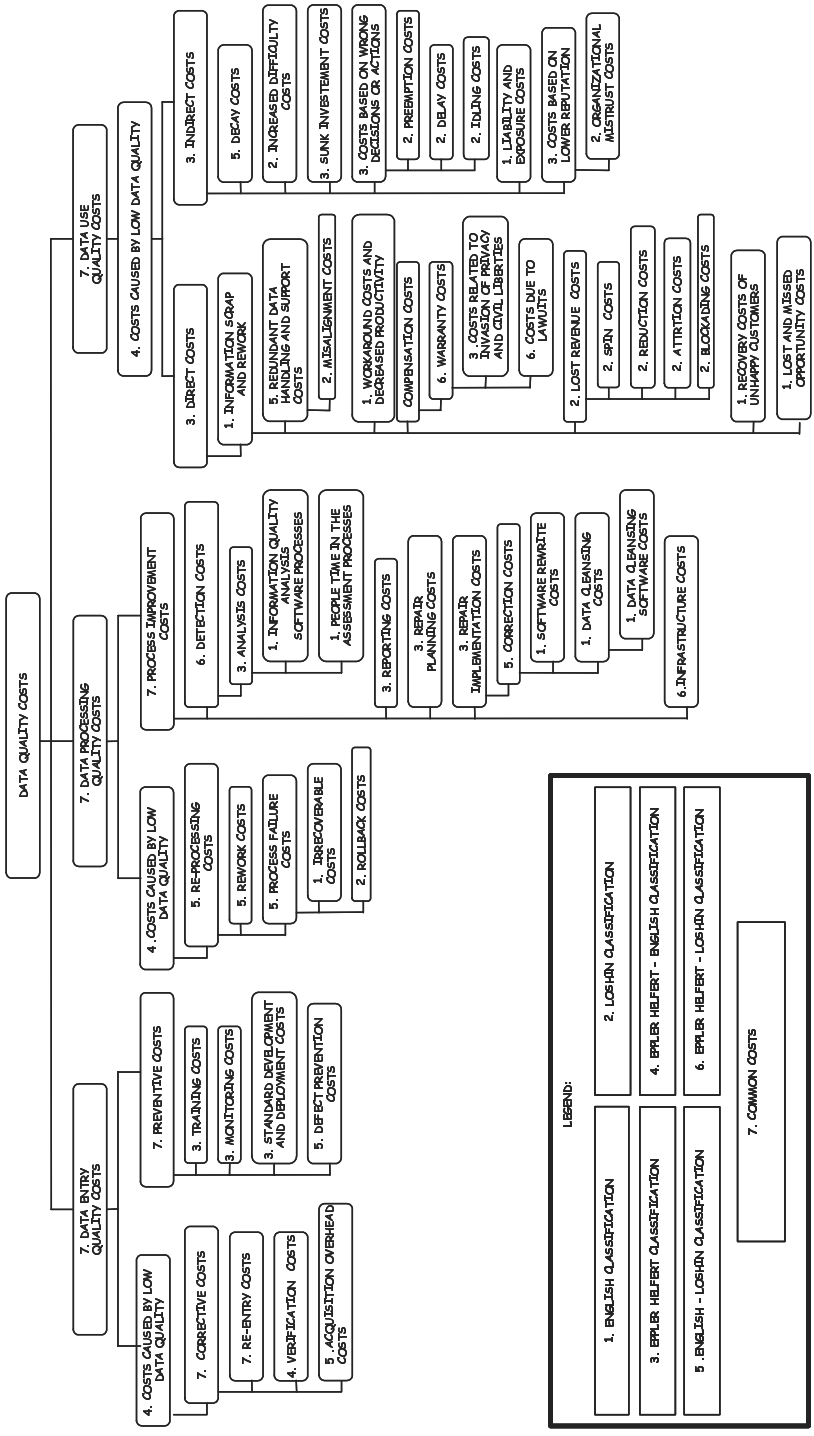



**Fig. 4.13.** A comparative classification for costs

### 4.4.2 Benefits Classification

Benefits can typically be classified into three categories:

1. *Monetizable*, when they correspond to values that can be directly expressed in terms of money. For example, improved data quality results in increased monetary revenues.
2. *Quantifiable*, when they cannot be expressed in terms of money, but one or more indicators exist that measure them, expressed in a different numeric domain. For example improved data quality in Government-to-Business relationships may result in reduced wasted time by businesses, which can be expressed in terms of a time indicator. Observe that in several contexts a quantifiable benefit can be expressed in terms of a monetizable benefit if a reasonable and realistic conversion function is found between the quantifiable domain and money. In our example, if the time wasted by business is productive time, the "wasted time" quantifiable benefit can be translated in terms of the monetizable benefit "unproductively spent money."
3. *Intangible*, when they cannot be expressed by a numeric indicator. A typical intangible benefit is the loss of image of an agency or a company due to inaccurate data communicated to customers, e.g., requests to citizens for undue tax payments from the revenue agency.

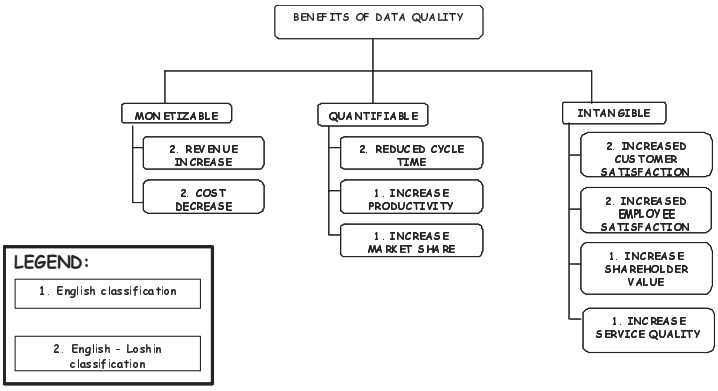



**Fig. 4.14.** A comparative classification for benefits

Figure 4.14 shows the English and Loshin items represented together, corresponding to benefits in the three categories. With regard to monetizable benefits, the two classifications agree in the indication of economic issues related to revenue increase and cost decrease, while in quantifiable and intangible benefits the English classification is richer; among the intangible benefits, the reference to service quality is relevant. In Chapter 7 we will see examples of applications of the above classifications in a real case study.

## 4.5 Summary

In this chapter we have introduced several data quality activities, discovering that the improvement of data quality in an organization can be performed with a variety of actions and strategies. All of the activities introduced apply to data, and produce data of improved quality according to a given process. Other improvement activities can rely on processes that manipulate data, modifying the process or introducing suitable controls in the process; we will discuss them in Chapter 7.

We have also started the discussion on activities while thoroughly analyzing (i) quality composition, and (ii) error localization and correction. Finally, we have discussed cost-benefit classifications in data quality, that can be used as check lists in the process of cost and benefit allocation. For quality composition and error localization and correction we introduced a spectrum of techniques for several possible cases, while for cost/benefit classifications we compared the different approaches. In such a way, we provided a framework for analysis that allows the reader to choose the specific approach to adopt based on the context of use.

# 5

# Object Identification

In this chapter, we describe object identification, probably the most important and the most extensively investigated data quality activity.

In order to introduce critical issues, and justify the structure of the chapter, let us describe an example related to an e-Government application scenario. In such a scenario, different agencies manage administrative procedures related to different types of businesses in order to register their information on businesses in their respective national registries, authorize specific activities, and provide services, e.g. for collecting taxes. In each agency, the same set of businesses is represented, with some attributes common and other attributes specific to the agency. We have reported in Figure 5.1 a real-life example of the same business as represented in three national registries (some details, irrelevant in this context, have been changed for privacy reasons).

| Agency | Identifier | Name | Type of activity | Address | City |
|---|---|---|---|---|---|
| Agency 1 | CNCBTB765SDV | Meat production of John Ngombo | Retail of bovine and ovine meats | 35 Niagara Street | New York |
| Agency 2 | 0111232223 | John Ngombo canned meat production | Grocer's shop, beverages | 9 Rome Street | Albany |
| Agency 3 | CND8TB76SSDV | Meat production in New York state of John Ngombo | Butcher | 4, Garibaldi Square | Long Island |

**Fig. 5.1.** How three agencies see the same business

The three tuples present several differences:

1. Values of the identifiers are different due to different policies of the three agencies; also, in the case in which they share a common domain and meaning (this is the case for Agencies 1 and 3), they differ due to some data entry errors.

2. Names are different, although several common or similar parts exist (also, in this case, some data entry error can be recognized).
3. Types of activity are different; this difference may be due to several reasons, such as typos, deliberately false declarations, or data updated at different times.
4. Further differences appear in remaining `Address` and `City` attributes.

Yet, the three tuples represent the same business!

We call *object identification* the data quality activity needed to identify whether data in the same source or in different ones represent the same object of the real world.

As mentioned in Chapter 1, poor data quality in a single database produces poor service quality and economic losses. Poor data quality referring to the same types of objects (e.g., persons, businesses and portion of territory) in different databases yields poor results in all applications (e.g., queries, transactions and aggregations) that access the same objects in the different databases. This type of access is typical of many Government/Business/Citizen-to-Government/Business/Citizen interactions. For example, to discover tax frauds, different agencies can cross-check their databases in order to search for contradictions or correlations among data: this is possible only if data referring to the same object can be identified.

This chapter is organized as follows. In Section 5.1 we briefly provide a historical perspective of the object identification problem. In Section 5.2, we discuss the different data typologies involved in the object identification process. In Section 5.3 we describe the general steps of the process that are detailed in Section 5.4. In Section 5.5 we introduce the specific object identification techniques that are detailed in the following sections: Section 5.6 describes probabilistic techniques, Section 5.7 illustrates the empirical ones, and, finally, Section 5.8 details the knowledge-based techniques. The chapter ends with a comparison of the techniques in Section 5.9.

## 5.1 Historical Perspective

The term *record linkage* is mentioned for the first time in [64]. Since computer applications have been used to automate more and more administrative activities, demographic studies, health experiments, and epidemiological analyses, it has become clear that data often result from the merging of different sources, created and updated at different times and by different organizations or persons. Moreover, merging data produces new data of potentially higher value, since properties that are merged can be related with new types of aggregations, analyses, and correlations.

In 50′ and 60′, data was represented in *files*, *records*, and *fields*, and terminology that justifies the original term *record linkage* as the activity that results in the integration of information from two or more independent sources.

In this chapter we will frequently use the *file/record/field* terminology, instead of the *relation/tuple/attribute* terminology, whenever the techniques apply to the more general file structure.

One of the first efforts for moving from empirical procedures to formal methods originates from the geneticist Howard Newcombe [146], who introduced frequencies of occurrences of values in strings and decision rules for matching and non-matching records. Such procedures were used in the development of health files of individuals. Fellegi and Sunter [77] provided a mature formal theory for record linkage (see Section 5.6.1). A great number of subsequent experiments and theoretical improvements originated, in addition to health applications, also in administrative and census applications, characterized by a large amount of data, from sources with various degrees of trustworthiness and accuracy. In such applications, it is crucial to produce efficient computer-assisted matching procedures that can reduce the use of clerical resources, and effective methods that can reduce errors in matching and non-matching. See [216] for a general discussion on the peculiarities of record linkage methods on administrative data.

In recent years, new techniques have been proposed that extend the linkage activity from files to more complex structures. Such techniques also try to exploit knowledge on the application domain to produce more effective decision procedures. These topics will be examined in more detail in the following sections.

## 5.2 Object Identification for Different Data Types

Techniques developed for dealing with the object identification problem strictly depend on the type of data used to represent objects. Refining and adapting the classifications provided in Chapter 1, we distinguish three main data types that refer to the same class of objects:

1. *Simple structured data,* that correspond to pairs of files or relational tables.
2. *Complex structured data*, i.e., groups of logically related files or relational tables.
3. *Semi-structured data*, such as pairs of XML marked documents.

In Figure 5.2, data of the three different types are shown. In Figures 5.2a and 5.2b, an object of type `Person` is represented, while a `Country` is represented in Figure 5.2c.

In order to discover matching and non-matching objects within the three structures, we need intuitively different strategies. Historically, simple structured data correspond to traditional files, which have poor mechanisms to represent the semantics of data. With the advent of database management systems (DBMSs), and, specifically, relational DBMSs, it has been possible to assign semantics to such structures, in terms of domains, keys, functional dependencies, and constraints. The advent of networks and Internet and the

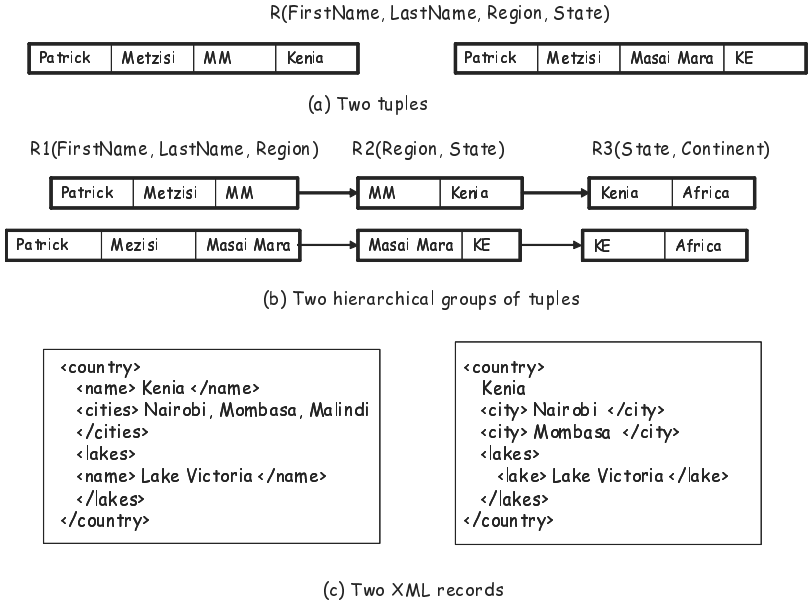



**Fig. 5.2.** Examples of matching objects of the three data typologies

development of the XML standard have pushed the investigation of techniques for semi-structured data.

In relation to the above discussion, two different terms are widely used in the literature: *record linkage* and *object identification*. Other terms used are *record matching* and *entity resolution*. *Record linkage* is used when the matching activity is performed on simple structured data, in our terminology, files or relations. Usually, it is known a priori that the two relations model the same entity of the real world, e.g. persons, businesses, or buildings. The goal of record linkage is to produce a new file where all the tuples of the two input files referring to the same entity of the real world (e.g., the same person, the same business) are merged into a unique record; techniques may also simply produce the cluster of matching records without choosing the representative record. When a unique file is considered, the goal of record linkage is to discover and unify the records in the file that refer to the same entity of the real world; in this case, it is called *deduplication* or *duplicate identification*.

*Object identification* is an evolutive term for record linkage, and deals with complex structured data and XML documents where objects of the real world are represented, in general, with a wider spectrum of structures than simple structured data. For instance,

1. in data warehouses, objects used for dimensions in a star schema are represented with a group of relations related by foreign key constraints; this is the case of the tuples in Figure 5.2b;
2. in normalized relational schemas, several relations are needed to represent an object; and
3. in documents, objects are hidden in natural language descriptions, and their presence may be revealed by some schema specification (e.g., XML schemas).

These characteristics call for more sophisticated techniques when moving from simple structured data to complex structured data and semistructured data; at the same time, the semantic wealth of DBMS and XML models, in comparison to files, provides richer mechanisms (e.g., keys) to reveal structural similarities between data, resulting in more complex, but also more powerful, techniques.

## 5.3 The High-Level Process for Object Identification

Although inspired by different general paradigms and tailored to the different types of data introduced in the previous section, techniques for object identification have a generally common structure, described with different levels of detail in Figures 5.3 and 5.4, where we assume for simplicity we have two files as input data.

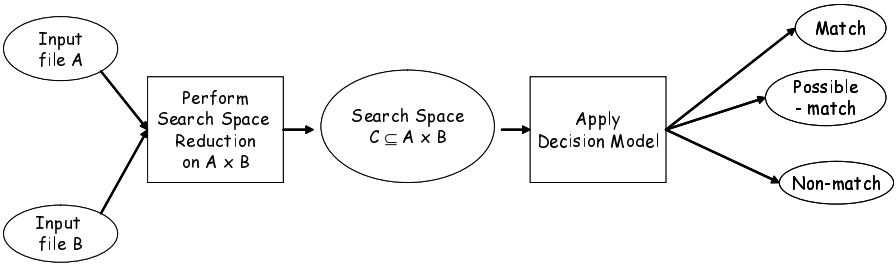



**Fig. 5.3.** Relevant steps of object identification techniques

In Figure 5.3, starting from the potential search space, consisting of the cartesian product of tuples in input files, a reduced search space is first constructed. The reason for this step is to reduce the complexity of the technique, which, otherwise, is $O(n^2)$, where $n$ is the cardinality of each of the input relations. Then, a decision model is used to decide if records in the reduced search space match, i.e., correspond to the same object, do not match, or no decision can be made automatically, and a domain expert has to be involved. Minimization of *possible matches* is a typical goal of object identification techniques to reduce clerical involvement. At the same time, a further goal to be

achieved is to minimize *false positives*, i.e., false assignments of pairs of tuples to the match decision, and the complementary *false negatives*.

**Step 1. Preprocessing**
Standardize fields to compare and correct simple errors
**Step 2. Perform search space reduction**
Given the search space A x B of the two files, find a new search space C ⊆A X B to apply further steps
**Step 3. Choose comparison function**
Choose the function(s)/set of rules that expresses the distance between records in C
**Step 4. Apply decision model**
Choose the method for assigning pairs in C to M, the set of matching records, U the set of unmatching records, and P the set of possible matches
**Step 5. Verification**
Check the effectiveness of method if not satisfactory, go back to Step 2

**Fig. 5.4.** Description of relevant steps

Figure 5.4 adds three more phases to the general process, namely,

- a *prepropressing* activity that has the goal of working on data in order to standardize it and correct evident errors (see Section 5.4.1);
- the *choice of a comparison function* between tuples, to be used in the decision model activity;
- a *verification* step, during which some quality measures are performed to assess if the result is satisfactory, and, if needed, to iterate the method, such as by making a different choice (for example, adopting a new comparison function).

Three major categories of techniques for object identification can be identified on the basis of the underlying research paradigms:

1. *Probabilistic techniques*, based on the extremely relevant set of methods developed in the last two centuries in statistics and probability theory, ranging from Bayesian networks to data mining tools.
2. *Empirical techniques* that make use in the different phases of the process of algorithmic techniques such as sorting, tree traversal, neighbor comparison, and pruning.
3. *Knowledge-based techniques*, in which domain knowledge is extracted from the files involved, and reasoning strategies are applied to make the process more effective.

Both in probabilistic and in knowledge-based techniques, the steps of the general procedure, described in Figure 5.4, can be either performed independently of the domain (*domain-independent techniques*) or could be based on domain-specific information or knowledge (*domain-dependent techniques*).

Furthermore, in some applications it is useful to have a priori a sample of data for which it is known whether they match or not; such a sample is called *labeled data*, while *unlabeled data* are data for which the matching status is

unknown. Labeled data can be used effectively to learn probabilities, distance functions, or knowledge used in the different techniques. Therefore, two different types of learning can be identified: *supervised learning*, when knowledge is available on matching/unmatching pairs, and *unsupervised learning*, when the source knowledge is of a different nature (e.g., integrity constraints on the domain).

Finally, in the case in which complex structured data and semistructured data are involved, further tree/graph traversal activity is needed in order to apply the strategy to all parts of the structure.

## 5.4 Details on the Steps for Object Identification

In this section, the first three steps described in Figure 5.4, namely, preprocessing, search space reduction, and the issues related to comparison functions, will be illustrated in detail. The next sections deal with step 4, apply decision method. In the last section of the chapter we will introduce metrics for step 5, verification.

### 5.4.1 Preprocessing

The preprocessing step includes the following activities:

- *Conversion of upper/lower cases*, in which data to be compared corresponding to alphabetic strings are transformed to be homogeneous in terms of upper and lower cases. So, for instance, if names of companies are stored such that the first character is upper case, then the corresponding strings are converted such that all their characters are lower cases, e.g. `Hewlett Packard` is transformed into `hewlett packard`,and `Microsoft` into `microsoft`.
- *Replacement of null strings*. Null strings must be replaced in order to allow proper comparisons. For example, `hewlett packard` must be transformed into `hewlettpackard`.
- *Standardization*, consisting of reorganization of composed fields, data type checks, replacement of alternative spellings with a single one. A typical example of *reorganization* of a composite field is given by addresses. In many applications addresses are stored as a single string; the standardization activity may consider parsing the string into substrings corresponding, for instance, to `StreetName`, `CivicNumber`, `City`, and `State`. In the context of object identification, this type of reorganization has the purpose of making comparisons easier. However, it can be performed also to facilitate accuracy checks. Indeed, for fields derived from decomposition, dictionaries may be available for the use as lookup tables for correcting the data. *Data type checks* regard the standardization of formats. For example, dates must be expressed in the same format: 1 Jan 2001, 01-1-2001, 1st January

2001 should be homogenized to a single format. *Replacement of alternative spellings* include abbreviations that can be replaced by the corresponding complete word, e.g., `rd.` by `road`.

- *Schema reconciliation* is a more complex activity that must address all conflicts that can occur when data under consideration come from disparate data sources. Examples of such conflicts are heterogeneity conflicts, semantic conflicts, description conflicts, and structural conflicts. More details on this can be found in Chapter 6.

### 5.4.2 Search Space Reduction

The object identification problem has a search space dimension equal to the cardinality of $\mathtt{A} \times \mathtt{B}$, given two sets of records `A` and `B` to be compared. The reduction of the search space can be done by three different methods, namely, blocking, sorted neighborhood and pruning (or filtering).

*Blocking* implies partitioning a file into mutually exclusive blocks, and limiting comparisons to records within the same block. Blocking can be implemented by choosing a *blocking key* and grouping into a block all records that have the same values on the blocking key. Blocking can also be implemented by *hashing*. The blocking key is used for hashing records in hash blocks. If $b$ is the number of blocks and $n/b$ is the dimension of each block, then the total time complexity of blocking is $O(h(n) + n^2/b)$ where $h(n)=n \log n$ if blocking is implemented by sorting, or $h(n)=n$ if blocking is implemented by hashing.

*Sorted neighborhood* consists of sorting a file and then moving a window of a fixed size on the file, comparing only records within the window. The number of comparisons is consequently reduced from $n^2$ to $O(wn)$, where $w$ is the size of the window; considering the sorting complexity $O(n \log)$, the method requires a total time complexity of $O(n \log n + wn)$. See also Section 5.9.2 for a comparison between blocking and sorted neighborhood methods.

*Pruning* (or *filtering*) has the objective of first removing from the search space all records that cannot match each other, without actually comparing them. As an example, let us consider the case where two records are declared to be a match if a given comparison function $f(r_i, r_j)$ is greater than a threshold $\tau$. If an upper bound for $f$ is found, e.g., $f(r_i, r_j) <= \delta(r_i)$ for each $j$, then, if $\delta(r_i) <= \tau$, $f(r_i, r_j)$ will be lower than $\tau$ for each $r_j$; therefore, $r_i$ cannot have any record to be matched with, and can be removed from the search space.

### 5.4.3 Comparison Functions

Comparison functions have been widely investigated, especially string comparison functions (see surveys [90] and [143]). In the rest of this section, we review some of the most important functions, and we provide examples to show similarities and differences.

*Edit distance.* The edit distance between two strings is the minimum cost of converting one of them to the other by a sequence of character insertions,

deletions, and replacements. Each of these modifications is assigned a cost value. As an example, assuming that the insertion cost and the deletion cost are each equal to 1, the edit distance between the two strings `Smith` and `Sitch` is 2, as `Smith` is obtained by adding `m` and deleting `c` from `Sitch`.

*n-grams, bi-grams, q-grams.* The n-grams comparison function forms the set of all the substrings of length $n$ for each string. The distance between the two strings is defined as: $\sqrt{\sum_{\forall x} |f_{s'} - f_{s''}|}$, where $f_{s'}$ and $f_{s''}$ are the number of occurrences of the substrings $x$ in the strings $s'$ and $s''$, respectively. Bi-grams comparison ($n = 2$) is widely used, and is effective with minor typographical errors. Positional q-grams are obtained by sliding a window of length $q$ over the characters of a string $s$.

*Soundex code.* The purpose of the soundex code is to cluster together names that have similar sounds. For example, the soundex code of `Hilbert` and `Heilbpr` is similar. A soundex code always contains four characters. The first letter of the name becomes the first character of the soundex code. The remaining three characters are drawn from the name sequentially, by accessing a predefined table. As an example, the soundex code of `Hilbert` and `Heilbpr` is H416. Once the four-character limit has been reached, all remaining letters are ignored.

*Jaro algorithm.* Jaro introduced a string comparison function that accounts for insertions, deletions, and transpositions. Jaro's algorithm finds the number of common characters and the number of transposed characters in the two strings. A *common character* is a character that appears in both strings within a distance of half the length of the shorter string. A *transposed character* is a common character that appears in different positions. As an example, comparing `Smith` and `Simth`, there are five common characters, two of which are transposed. The (scaled) Jaro string comparator is given by

$$f(s_1, s_2) = \frac{\frac{N_c}{lengthS_1} + \frac{N_c}{lengthS_2} + 0.5\frac{N_t}{N_c}}{3},$$

where $s_1$ and $s_2$ are strings of lengths $lengthS_1$ and $lengthS_2$ respectively, $N_c$ is the number of common characters between the two strings (where the distance for common characters is half the minimum length of $s_1$ and $s_2$), and $N_t$ is the number of transpositions.

*Hamming distance.* The Hamming distance counts the number of mismatches between two numbers. It is used primarily for numerical fixed size fields like zip codes or social security numbers. For example, the Hamming distance between 00185 and 00155 is 1 because there is one mismatch.

*Smith-Waterman.* Given two sequences, the Smith-Waterman algorithm uses dynamic programming to find the lowest cost of changes that convert one string into another. Costs for individual changes, namely modifications, insertions, and deletions, are parameters of the algorithm. The algorithm performs well for many abbreviations, taking into account gaps of unmatched characters, and also when records have missing information or typographical mistakes.

*TF-IDF.* The Token Frequency-Inverse Document Frequency (TF-IDF) or *cosine similarity* is widely used for matching similar strings in documents. The basic idea is to assign higher weights to tokens appearing frequently in a document (TF weight) and to assign lower weights to tokens that appear frequently in the whole set of documents (IDF weight). For a term $i$ in a document $j$ the weight $w_{i,j}$ is

$$w_{i,j} = (tf_{i,j}) \times \log(\frac{N}{df_i})$$

where $tf_{i,j}$ is the number of occurrences of $i$ in $j$, $df_i$ is the number of documents containing $i$, and $N$ is the total number of documents. The similarity between two documents is then computed as the cosine between their respective weighted term vectors. Specifically, being `V` $= \{w_1, \ldots, w_n\}$ and `U` $= \{w_1, \ldots, w_n\}$ the weighted term vectors, the cosine similarity is

$$\frac{V \cdot U}{\mid V \mid \cdot \mid U \mid}.$$

## 5.5 Object Identification Techniques

In Figure 5.5, the set of object identification techniques that will be detailed in the rest of this chapter is shown. Each technique is described by a name, the technical area within which the technique was proposed (probabilistic, empirical, or knowledge-based) and the type of data representing objects to be identified (pairs of files, relational hierarchies, or XML documents). Several object identification techniques are not described in the text, including [45, 62, 172] and [115]. The main criteria used to select the listed techniques are

- adoption: Fellegi and Sunter (and its extensions) is the first and by far the more established technique, and it is representative of probabilistic techniques. The sorted neighborhood method and its variants are also representative of empirical methods.
- novelty: DogmatiX is among the first techniques actually dealing with object identification in XML documents, and Delphi is among the first ones dealing with complex structured data. Cost-based techniques have the originality of dealing with costs of linkage errors. Both the knowledge-based techniques are actually novel contributions, as there are quite a few works on knowledge-based approaches to object identification.

## 5.6 Probabilistic Techniques

In this section we describe the probabilistic techniques based on the Fellegi and Sunter theory, providing the original model, subsequent extensions, and a cost-based technique.

| Name | Technical Area | Type of data |
|---|---|---|
| Fellegi and Sunter and extensions | probabilistic | Two files |
| Cost-based | probabilistic | Two files |
| Sorted Neighborhood and variants | empirical | Two files |
| Delphi | empirical | Two relational hierarchies |
| DogmatiX | empirical | Two XML documents |
| Intelliclean | knowledge-based | Two files |
| Atlas | knowledge-based | Two files |

**Fig. 5.5.** Object identification techniques

### 5.6.1 The Fellegi and Sunter Theory and Extensions

The record linkage theory was proposed by Fellegi and Sunter in [77]. In this section, we summarize the proposed theory and briefly describe the subsequent extensions and refinements.

Given two sets of records A and B, let us consider the cross product $\mathtt{A} \times \mathtt{B} = \{(\mathtt{a}, \mathtt{b}) | \mathtt{a} \in \mathtt{A}$ *and* $\mathtt{b} \in \mathtt{B}\}$. Two disjoint sets M and U can be defined starting from $\mathtt{A} \times \mathtt{B}$, namely, $\mathtt{M} = \{(\mathtt{a}, \mathtt{b}) | \mathtt{a} \equiv \mathtt{b}, \mathtt{a} \in \mathtt{A}$ *and* $\mathtt{b} \in \mathtt{B}\}$ and $\mathtt{U} = \{(\mathtt{a}, \mathtt{b}) | \mathtt{a} !\equiv \mathtt{b}, \mathtt{a} \in \mathtt{A}$ *and* $\mathtt{b} \in \mathtt{B}\}$, where the symbol $\equiv$ means that the records a and b represent the same real world entity (and $!\equiv$ they do not). M is named the *matched set* and U is named the *unmatched set*. The record linkage procedure attempts to classify each record pair as belonging to either M or U. A third set P can be also introduced representing possible matches.

Let us suppose that each record in A and B is composed of $n$ fields; a *comparison vector* $\gamma$ is introduced that compares field values of records $\mathtt{a}_i$ and $\mathtt{b}_j$ (see Figure 5.6), namely, $\gamma = [\gamma_1^{ij}, \ldots, \gamma_n^{ij}]$. $\gamma$ is obtained by means of comparison functions, defined as $\gamma_k^{ij} = \gamma(\mathtt{a}_i(k), \mathtt{b}_j(k))$, denoted in the following for brevity as $\gamma_k$. Usually, only a subset of the fields of A and B is compared. $\gamma$ is a function of the set of all $\mathtt{A} \times \mathtt{B}$ record pairs; with each couple of fields of each pair, it associates a specific level of agreement. As an example, given two files with fields `Name`, `Surname`, and `Age`, we may define a $\gamma$ comparison function made of three predicates on each of the fields, namely `agree Name`, `agree Surname`, and `agree Age`.

The functions $\gamma_i$ can compute a binary agreement on values, i.e., $\gamma(\mathtt{v}_1, \mathtt{v}_2) = 0$ if $\mathtt{v}_1 = \mathtt{v}_2$, and 1 otherwise; the functions can also compute a three-value result, i.e., $\gamma(\mathtt{v}_1, \mathtt{v}_2) = 0$ if $\mathtt{v}_1 = \mathtt{v}_2$, 1 if either $\mathtt{v}_1$ or $\mathtt{v}_2$ is missing, 2 otherwise. The functions can also compute continuous attribute values; relevant comparison functions are described in detail in Section 5.4.3. The set of all comparison vectors is the comparison space $\Gamma$.

Given $(\mathtt{a}_i, \mathtt{b}_j)$, the following conditional probabilities can be defined:

- $m(\gamma_k) = \Pr(\gamma_k | (\mathtt{a}_i, \mathtt{b}_j) \in \mathtt{M})$ and

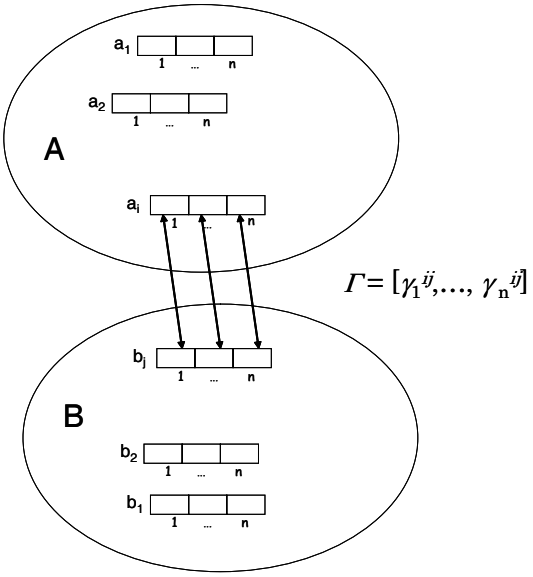



**Fig. 5.6.** The Fellegi and Sunter record linkage formulation

- $u(\gamma_k)$=Pr($\gamma_k|(\mathtt{a}_i, \mathtt{b}_j) \in \mathtt{U}$).

As an example, for the above files with fields `Name`, `Surname`, and `Age`, the probabilities `Pr(agree Name|M), Pr(agree Surname|M), and Pr(agree Age|M)` and `Pr(agree Name|U), Pr(agree Surname|U), and Pr(agree Age|U)` can be defined. Note that the size of $\Gamma$ depends on its inner structure.

By considering all the fields, we define analogous formulas for $\gamma$:

- $m(\gamma)$=Pr($\gamma|(\mathtt{a}_i, \mathtt{b}_j) \in \mathtt{M}$) and
- $u(\gamma)$=Pr($\gamma|(\mathtt{a}_i, \mathtt{b}_j) \in \mathtt{U}$).

The above probabilities are called *m-* and *u-probabilities*, respectively. In the case in which we are able to estimate such probabilities, they become crucial in a possible assignment decision procedure. Fellegi and Sunter introduced the ratio R among such probabilities as a function of $\gamma$, namely,

$$R = m(\gamma)/u(\gamma),$$

where $\gamma$ ranges in the comparison space $\Gamma$, and, we recall, is a function of the set of all $\mathtt{A} \times \mathtt{B}$ record pairs. The ratio $R$, or the natural logarithm of such a ratio, is called *matching weight*. By composition, $R$ is a function of the set of all $\mathtt{A} \times \mathtt{B}$ record pairs.

Fellegi and Sunter defined the following *decision rule*, where $T_\mu$ and $T_\lambda$ are two thresholds (on them we will comment in a moment):

- if $R > T_\mu$, then designate pair as a match,

- if $T_\lambda <= R <= T_\mu$ then designate pair as a possible match,
- if $R < T_\lambda$ then designate pair as a non-match.

The area $T_\lambda <= R <= T_\mu$ partitions the set of $\gamma \in \Gamma$, and corresponding record pairs, into three disjoint subareas, namely, $A_1$, including pairs declared as *match*, $A_2$, including pairs declared as *possible match*, and $A_3$, including pairs declared as *non-match*. Figure 5.7 shows the three areas, where record pairs in the areas (represented with pairs of white and gray circles) are ordered to be monotonically decreasing by matching weight $R$. The figure shows that pairs designated as matching are usually much less than pairs designated as non-matching.

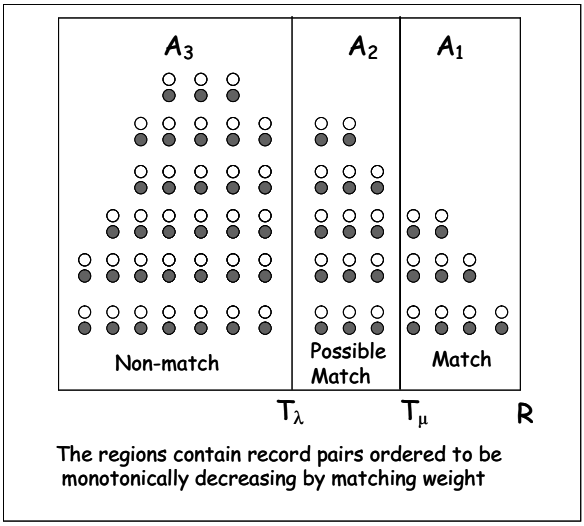



**Fig. 5.7.** The three areas of pairs defined by the decision rule

It is clear that the thresholds $T_\mu$ and $T_\lambda$ play an essential role in the decision procedure. Therefore, an important problem is how to fix them. Observe that if $\gamma$ consists mainly of agreements, then $R$ is large; conversely if $\gamma$ consists mainly of disagreements $R$ is small. Since $R$ is a ratio of probabilities, the assignment of pairs (`a`, `b`) for each value of $R$ to the matching set `M` or to the non-matching set `U` may results in possible false assignments. *False matches* and *false non-matches* are the two types of errors that are possible in the model, and $\mu$ and $\lambda$ represent the related error rates. High values of $R$ (see the $A_1$ area in Figure 5.8) correspond to low probability of false matches assignments, with the probability of false matches increasing while values of $R$ decrease. Similarly, for low values of $R$, the probability of false non-match decreases while decreasing the values of $R$. In Figure 5.8, the line crossing the three areas represents a possible trend of probabilities of false matches and false non-matches. So, the three areas are identified by specific values of $T_\lambda$

and $T_\mu$, and the $A_1$ and $A_3$ regions are further divided into true/false match and true/false non-match regions, respectively.

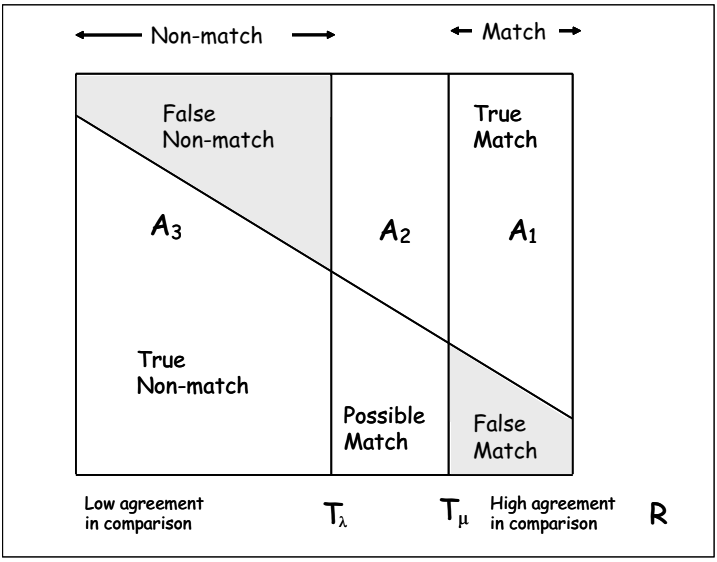



**Fig. 5.8.** The regions of the Fellegi and Sunter Model [88]

In order to provide criteria to fix the two thresholds $T_\mu$ and $T_\lambda$, we have to decide which are the rates of error we are willing to accept in the decision rule proposed above; such error rates correspond to the two gray areas in Figure 5.8. Once the error rates are fixed, the two thresholds are consequently fixed. Fellegi and Sunter proved that the above decision rule is optimal, where optimal means that the rule minimizes the probability of classifying pairs as belonging to the area $A_2$ of possible matches.

### Parameters and Error Rates Estimation

The Fellegi and Sunter theory is based on the knowledge of the $u$- and $m$-probabilities. Several methods have been proposed to compute or estimate such probabilities. First, Fellegi and Sunter proposed a method to compute the $u$- and $m$-probabilities providing a closed-form solution under certain assumptions. More specifically, considering that

$$\Pr(\gamma) = \Pr(\gamma|\mathtt{M})\Pr(\mathtt{M}) + \Pr(\gamma|\mathtt{U})\Pr(\mathtt{U})$$

they observed that if the comparison vector $\gamma$ regards three fields, among which a conditional independence assumption holds, then a system of seven equations and seven unknowns can be solved to find $\Pr(\gamma|\mathtt{U})$ and $\Pr(\gamma|\mathtt{M})$

(being $7 = 2^3 - 1$, where the subtracting term is due the constraint that probabilities must be equal to 1).

Several parameter estimation methods for the theory have been proposed in the literature. Basically, such methods provide an *estimation* of the $u$- and $m$-probabilities rather than a computation of such parameters in closed form. The expectation-maximization algorithm and machine learning methods are the principal methods used for the estimation.

The *expectation-maximization*(EM) algorithm is used to find maximum likelihood estimates of parameters in probabilistic models, where the model depends on unobserved latent variables. EM includes an expectation (E) step, which computes the expected values of the latent variables, and a maximization (M) step, which computes the maximum likelihood estimates of the parameters, given the data and setting the latent variables to their expectation [61].

While continuing holding the conditional independence assumption, Winkler first showed how to estimate $m$- and $u$- probabilities by means of the EM-algorithm in [211]. Jaro [105] proposed another method to compute the $m(\gamma), \gamma \in \Gamma$ with the EM algorithm, which is implemented by commercially available software. Estimation methods have focused more recently on specific domains, such as persons and businesses, and specific fields, such as first names, last names, street names (see [212] for a detailed discussion).

The conditional dependence assumption holds very rarely. Proposals for estimating $m$- and $u$-probabilities under the *conditional dependence assumption* have been made in various works that come from the areas of statistics, information retrieval, and machine learning (see [214] for a survey). Specifically, generalized EM methods can be used ([210]) for estimations of such probabilities. The methods of Larsen and Rubin [113] are based on Bayesian models. The probability estimation of such methods are not accurate enough to estimate the error rates in the record linkage. The proposal of Belin and Rubin [23] goes in the direction of addressing this limitation. Specifically, Belin and Rubin proposed a mixture model for estimating false match rates, for given threshold values. The method requires training data and works well in a few situations, i.e., when there is a good separation between weights for matching and non-matching. Also, training data are considered a problem with very large data files.

In machine learning applications, typically, labeled training data (see section 5.3) are used, for which the true classification is known, allowing *supervised learning*. In [147], it is observed that the use of Bayesian networks makes it possible to straightforwardly combine labeled and unlabeled data during training, in order to obtain suitable decision rules. If only unlabeled data are used, then the decision rules may be very poor.

### 5.6.2 A Cost-Based Probabilistic Technique

In this section we describe a probabilistic technique [197] for performing record matching with the aim of minimizing the cost associated with misclassification errors, corresponding to false matches and false non-matches in Figure 5.8.

As previously described, the Fellegi and Sunter model proves that the proposed decision rule is optimal with respect to the minimization of the area needing clerical review (possible matches), for any pair of fixed thresholds on the probabilities of false matches and false non-matches.

The perspective adopted in [197] is different, in that it aims to minimize the *cost* associated with the misclassification error. The cost is considered as constituted by two different components, namely, (i) the cost of the decision process, including, for instance, the number of comparisons needed for the decision, and (ii) the cost of the impact of a certain decision. The comparison vector that, as introduced, corresponds to the attribute values of two given records that need to be compared is indicated by $\bar{x}$. In the following we provide an example showing the difference between error-based models and cost models. Given a comparison vector (1,1,0) with the probability of 75% of appearing among matches and 25% of appearing among non-matches, a rule based on the minimum error would assign it to `M`. Conversely, assuming that the cost of misclassifying a record as a match is more than three times the cost of misclassifying a record as a non-match, the comparison vector would be assigned to `U`.

Costs are domain dependent and are considered given in the proposed model. Moreover, the matching probabilities of the comparison records are also considered as given. Given such inputs, the model produces as outputs the decision rule on the membership to `M` or `U` and the required thresholds.

In the model, the costs $c_{ij}$ are considered, meaning the costs of making a decision $A_i$ when the compared pairs of records has an actual matching status $j$ (`M` or `U`). Decisions correspond to assignments to the three areas $A_1$, $A_2$, and $A_3$ defined in Section 5.6.1, related, respectively, to matching, possible matching, and non-matching pairs. Therefore, a cost is assigned to each decision, as shown in the table in Figure 5.9.

| Cost | Decision | Actual Matching |
|---|---|---|
| $C_{10}$ | $A_1$ | M |
| $C_{11}$ | $A_1$ | U |
| $C_{20}$ | $A_2$ | M |
| $C_{21}$ | $A_2$ | U |
| $C_{30}$ | $A_3$ | M |
| $C_{31}$ | $A_3$ | U |

**Fig. 5.9.** Costs corresponding to various decisions

The cost that has to be minimized is given by

$$c_m = c_{10} * \mathrm{P}(d = A_1, r = \mathtt{M}) + c_{11} * \mathrm{P}(d = A_1, r = \mathtt{U}) + c_{20} * \mathrm{P}(d = A_2, r = \mathtt{M}) + c_{21} * \mathrm{P}(d = A_2, r = \mathtt{U}) + c_{30} * \mathrm{P}(d = A_3, r = \mathtt{M}) + c_{31} * \mathrm{P}(d = A_3, r = \mathtt{U}),$$

where $d$ is the predicted class of a pair of records and $r$ is the actual matching status of a pair of records. The attribution of every point in the decision space constituted by the union of $A_1$, $A_2$, and $A_3$ is done in order to have the cost $c_m$ minimized. Inequalities are imposed on a particular expression of $c_m$ obtained by applying the Bayes theorem and a few other transformations to the formulation given above. Further details can be found in [197].

## 5.7 Empirical Techniques

The first proposal for a record matching technique based mainly on an empirical approach can be traced to 1983, to the work by Bitton and DeWitt [28]. The idea is to detect *exact* duplicates in a table, first sorting the table and then checking the identity of neighboring tuples. This basic approach has been adapted and extended in subsequent works in order to detect *approximate* duplicates with the goal of achieving better accuracy and performance results. In this section, we will review some major empirical techniques, starting from the sorted neighborhood method (Section 5.7.1) and the related priority queue algorithm (Section 5.7.2), then describing a technique for matching complex structured data (Section 5.7.3), and concluding with a technique for matching XML data (Section 5.7.4) and some additional empirical approaches to search space reduction (Section 5.7.5).

### 5.7.1 Sorted Neighborhood Method and Extensions

The basic sorted neighborhood method (SNM) was proposed in [182] and [93], and is also referred to as the *merge-purge* method. Given a collection of two or more files, the sorted-neighborhood method is applied to a sequential list of records built from such files. The method can be summarized in three phases, depicted in Figure 5.10 (let $\mathtt{x}_i$, $\mathtt{y}_i$, and $\mathtt{z}_i$ denote a possible matching record $i$ in three different sources):

- *Create keys.* Given the list of records derived from the union of available sources in a single file (see Figure 5.10, left), a key is computed by extracting a subset of relevant fields or portions of fields. Indeed, the rationale is that similar data will have closely matching keys. If $N$ is the total number of records in the list, the complexity of this step is $\mathrm{O}(N)$.
- *Sort data.* On the basis of the key selected in the previous phase, records are sorted in the data list (see Figure 5.10, middle). The complexity of this step is $\mathrm{O}(NlogN)$.

- *Merge.* A fixed size window is moved through the sequential list of records, limiting the comparisons for matching records to those records in the window (see Figure 5.10, right). If the size of the window is $w$ records, then every new record entering the window is compared with the previous $w-1$ records to find matching records.The decision about matching records is made according to domain-specific rules expressed in *equational theory*. The complexity of the merging phase is $O(wN)$.

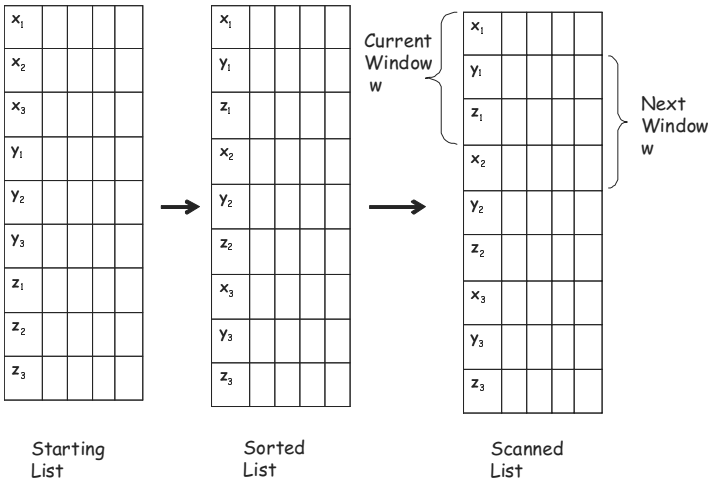



**Fig. 5.10.** Phases of the SNM method

When the three phases are applied serially, the total time complexity of the method is $O(NlogN)$ if $w < \lceil \log N \rceil$, $O(wN)$ otherwise.

In addition to the comparison performed in the merging phase, a *transitive closure step* is performed. Specifically, if records $\mathtt{r}_1$ and $\mathtt{r}_2$ are found to be similar, and records $\mathtt{r}_2$ and $\mathtt{r}_3$ are also found to be similar, then $\mathtt{r}_1$ and $\mathtt{r}_3$ are marked to be similar as well. Note that while the couples ($\mathtt{r}_1$, $\mathtt{r}_2$) and ($\mathtt{r}_2$, $\mathtt{r}_3$) must be within the same window to be declared as similar, the inferred similarity between ($\mathtt{r}_1$, $\mathtt{r}_3$) does not require the two records to belong to the same window. This property can be exploited in order to have smaller sizes for the scanning window, with invariant accuracy of the result.

The effectiveness of the sorted neighborhood method depends highly on the key selected to sort the records, since only keys of good quality cause similar records to be close to each other in the window, after the sorting phase. As an example, the first names of person records can be selected instead of last names, since we may suppose (or know) that last names can be more frequently misspelled than first names, which are typically more familiar. The SNM assumes that a "key designer" chooses the most suitable key, based

on considerations of the selectivity of the different attributes. In [26], the basic sorted neighborhood method is extended by making the choice of the key automatically. In order to choose a "good" key for matching instead of relying on "key designers", the idea is to rely on a quality characterization of records and on an identification power criterion that captures the selectivity of the different attributes. Experimental validation of the proposed method shows that whenever the quality characterization is taken into account, such automatic choice outperforms the basic SNM.

So far, the basic SNM has been described, running *once* on the list of concatenated source files. In the following, we describe two further versions: the *multi-pass approach*, which proposes several runs of the algorithm for more effectiveness, and the *incremental SNM*, which eliminates the need for the method to work on a single list of input data.

*Multi-pass Approach*

The multi-pass SNM is based on the consideration that running the SNM on a single sorting key does not produce the most suitable results. For example, if a highly selective key is chosen as the matching key, such as `SocialSecurityNumber`, even a single digit error can compromise the final result. Therefore, the idea is to have several runs of the method, each with a different key and very small windows. Having different keys allows to reasonably ensure that, if there are errors on some of them, the subsequent runs will compensate such errors. Also, running SNM with small windows is several less expensive steps instead of a single expensive one.

Each run of the multi-pass approach produces a set of pairs of records that can be merged. A transitive closure step is then applied to such pairs of records, and the result is the union of all pairs found in the independent runs, with the addition of pairs that can be inferred by transitive closure. The experimental evidence is that the multi-pass approach drastically improves the accuracy of the basic SNM with a single run on large varying windows, as also remarked in Section 5.9.

*Incremental SNM*

The incremental SNM is proposed for when it is too expensive to produce a single file of all input data. Typically, the step of producing a single file may be acceptable once; but, then, the problem occurs on how to deal with newly arrived data. The basic idea of the incremental SNM is to select a set of *prime-representatives* of records for each cluster deriving from the application of the SNM. Once new data need to be merged, they will be concatenated with the set of prime-representatives; the SNM will work on this concatenated set and new prime-representatives will be selected for subsequent incremental phases. Each cluster can have more than one eligible prime-representative, and the strategies for selecting them can be various. For example, a strategy could be to select the longest and most complete record. As another example, the

prime-representative could be selected as the record representing the most general concept within the cluster.

### 5.7.2 The Priority Queue Algorithm

The priority queue algorithm, first proposed in [134], is based on the same ideas of sorting and scanning as in the SNM. The main distinguishing issues are

- the usage of a domain-independent strategy to perform duplicate record detection, based on the Smith-Waterman algorithm [180] (see Section 5.4.3); and
- the usage of an efficient data structure, exploiting the union-find structure [187];
- the proposal of a heuristic method based on a priority queue for improving the performance of the SNM.

The *union-find* data structure is used for detecting and maintaining the connected components of an indirect graph. The problem of detecting duplicates can be modeled in terms of determining the connected components of a graph, if considering the transitivity of equality. Specifically, each record of the file can be modeled as a node of a graph, where an undirected edge connects two nodes if they match.

The matching of a pair of records can be recursively verified by considering if they belong to the same connected component: if they do, a match is declared; if they belong to different components a non-match is declared; otherwise, they are compared to each other, and, in case of a matching, a new component is added to the graph. The two operations of the union-find structure are *union* $(x,y)$, combining the set to which $x$ belongs with the set to which $y$ belongs (further, a representative for the union set is also chosen and the union set replaces the two initial sets); *find(x)*, returning the representative of the unique set containing $x$.

The algorithm considers a priority queue containing a fixed number of sets of records that are representatives of clusters. Only the most recently detected cluster members are stored in the queue. Given a record `a`, the algorithm first checks if it is a member of the clusters represented in the priority queue by comparing the cluster representative of `a` with the cluster representative of each set in the priority queue. This check is done by the *find* operation. If the check is successful, then `a` is already known to be a member of a cluster in the priority queue. If it is not successful, then `a` is compared with records in the priority queue by the Smith-Waterman algorithm. If a match is detected, the *union* function adds the `a`'s cluster to the cluster of the matched record; otherwise, `a` must be a member of a cluster not present in the queue, and so it is saved with the highest priority as a singleton set in the queue.

The priority queue algorithm can perform considerably better than SNM for very large files and databases. For instance, the number of record compar-

isons can be reduced up to five times for a database of 900,000 total records (see [134]). In Section 5.9, further details on the experimental results are provided.

### 5.7.3 A Technique for Complex Structured Data: Delphi

A technique for complex structured data is described in [7], where the Delphi algorithm is proposed; complex structured data considered in Delphi are called *dimensional hierarchies*; they consist of a chain of relations linked by foreign key dependencies. Given a pair of adjacent relations in the hierarchy, we call *parent* the relation on the foreign key side, and *child* the relation on the key side.

Dimensional hierarchies of relations are used typically (but not exclusively) in star schemas of data warehouses, where the chain of relations is composed of a relation representing the table of facts, and one or more relations representing the dimensions of interest for the multidimensional analysis, organized with various normalization degrees. We adopt in the following a more general term for dimensional hierarchies, namely, *relational hierarchies*.

An example of relational hierarchy is shown in Figure 5.11, where persons are represented in (i) the relation `Person`, (ii) their `Administrative Region` of residence (e.g., district or region, according to country), and (iii) `Country`. The relation `Country` is parent of the relation `Administrative Region` and is at the top of the hierarchy, while the relation `Person` is at the bottom. Note that `RegId` and `CtryId` are generated keys, used for an efficient link for pairs of tables.

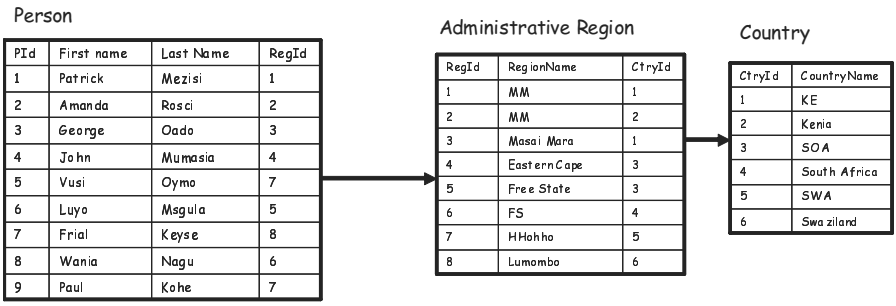



**Fig. 5.11.** Three hierarchical relations

In Figure 5.11, three different types of objects are represented in the schema:

1. persons, with region and country of residence;
2. regions, characterized by a set of resident persons and country;

3. countries, characterized by a set of regions and, for each region, a set of resident persons.

For each type of object, we may examine which are the duplicates in the relational hierarchy; e.g., we see that three different African countries are represented in the `Country` relation instance, with both the official name and an acronym.

The main idea of Delphi is to exploit the hierarchical structure of tuples, using both local (called *textual*) and global (called *co-occurrence*) similarity measures. Examine the tuples in the `Country` relation of Figure 5.11. If we simply adopt a similarity measure local to the relation, e.g., the edit distance between names of countries, we can falsely conclude that `<SOA, SWA>` are duplicates, and `<KE, Kenia>`, `<SOA, South Africa>`, `<SWA, Swaziland>` are not duplicates. If in addition to the edit distance we adopt a second distance that looks at how such items co-occur with linked tuples in the child `Administrative Region` relation, then we can see that (i) `KE` and `Kenia` have the `MM` tuple in common and (ii) for the three pairs `<KE, Kenia>`, `<SOA, South Africa>`, and `<SWA, Swaziland>` we can find non-overlapping groups of tuples linked with the pair.

The above example shows that in order to discover duplicates in relational hierarchies, we have to exploit the full structure of the hierarchy, or at least of adjacent relations. This strategy has two claimed advantages compared to "local" record linkage strategies:

1. it reduces the number of false matches, i.e., pairs of tuples incorrectly detected to be duplicates; this is the case with the pair `<SOA, SWA>`;
2. it reduces the number of false non-matches, i.e. pairs of tuples incorrectly detected as non duplicates; this is the case with the pair `<KE, Kenia>`.

More formally, traditional textual similarity measures are extended with a *co-occurrence similarity function* defined as follows. In a relational hierarchy, a tuple in a parent relation $\mathtt{R}_i$ joins with a set, which we call its children set, of tuples in the child relation; the co-occurrence between two distinct tuples is measured by the amount of overlap between the children sets of the two tuples. An unusually significant co-occurrence (more than the average overlap between pairs of tuples in $\mathtt{R}_i$ or above a certain threshold) is a cause for suspecting that one is a duplicate of the other. The above duplicate detection procedure can be performed for all types of objects represented in the hierarchy (in our example, persons, regions, and countries). Two objects are considered duplicates if corresponding pairs of tuples in each relation of the hierarchy either match exactly or are duplicates, according to duplicate detection functions at each level. The complete Delphi algorithm is described in Figure 5.12.

In order to make efficient the top-down traversal of the hierarchy and reduce the number of pairwise tuple comparisons, a potential duplicate identification filter is adopted to efficiently isolate a subset consisting of all potential

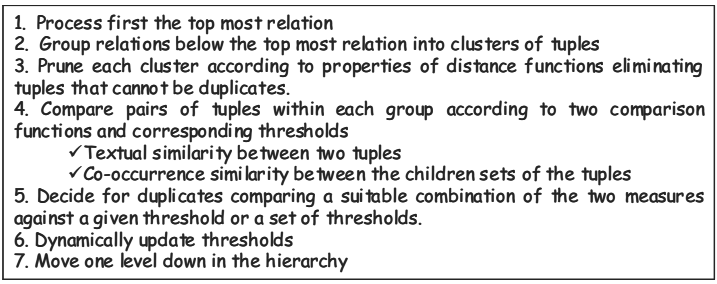



**Fig. 5.12.** The Delphi algorithm

duplicates, and prune the tuples that cannot be duplicates. The pruning step corresponds to step 2, state space reduction in Figure 5.4.

The dynamic threshold update step has the goal of adapting thresholds used in step 5 to structural characteristics of different groups; the number of items of the definition domain may vary across groups, and names of regions in one country may be longer or constitute a wider set than they are in another country, thus influencing the thresholds. See Section 5.9.4 on decision methods comparison.

### 5.7.4 XML Duplicate Detection: DogmatiX

In this section we describe a technique for object identification for XML documents. Finding duplicates in XML data has two major additional challenges when compared to files or relational data, namely, (i) the identification of objects to compare, and (ii) the possibility that the same elements are defined with different structures due to the flexibility of XML as a semistructured data model. In [207], an algorithm called DogmatiX (Duplicates Objects Get Matched in XML) that explicitly considers these features is proposed. The algorithm has a preprocessing phase that consists of three steps:

- *Step 1: candidate query formulation and execution.* XML data are first queried to extract duplicate candidates. Duplicate candidates are considered with respect to a real-world type. For instance, `Person` and `People` can be considered as two representations of the same real-world type `Individual`. Currently, the candidate selection is not done automatically in DogmatiX.
- *Step 2: description query formulation and execution.* The descriptions of duplicate candidates are expressed by queries that select only some of the properties that are associated with objects, namely, the ones that are considered meaningful for object identification. As an example, while the `Name` and `Surname` of a `Person` can be considered as relevant for identifying it, information about the person's hobbies cannot be relevant to the scope. Two heuristics to determine the candidate's descriptions are proposed in

[207]. The heuristics are based on a locality principle: given an element `e` the farther some information is from `e`, the less related it is to it.

- *Step 3: object description (OD) generation.* A relation consisting of tuples `OD(value, name)` is generated, where `value` describes an instance of some information and `name` identifies the type of information by name. For instance, `(Smith, Surname)` is part of the object descriptor for a `Person` instance included in the duplicate candidates.

After such a preprocessing phase, three actual steps for duplicate detection are performed:

- *Step 4: comparison reduction.* First a filter is applied to reduce the number of duplicate candidates: the filter is defined as an upper bound to the similarity measure and does not require the computation of such a measure, but preliminarily removes objects from the set of possible duplicates. Then, a clustering phase is applied in order to compare only objects within the same cluster.
- *Step 5: comparisons.* Pairwise comparisons are performed on the basis of a similarity measure. Such a similarity measure is defined in a domain-independent way (see [207] for details). The similarity measure takes into account some important features like (i) relevance of data or their identification power, by means of the introduction of a variant of the inverse document frequency (IDF) metric; (ii) the distinction between nonspecified and contradictory data; e.g., the fact that two persons have several different preferences may be an indicator the two persons are distinct, while a missing preference should not penalize the similarity measure.
- *Step 6: duplicate clustering.* The transitivity of the relationship *is-duplicate-of* is applied to XML objects selected as duplicates in Step 5.

The algorithm is a representative example of object identification for semistructured data.

### 5.7.5 Other Empirical Methods

The time efficiency of the record linkage process can be improved by the reduction of the search space, which can be performed by means of blocking and windowing strategies. For instance, instead of making detailed comparisons of all 10 billion pairs from two sets of 100,000 records representing all persons in a `State`, it may be sufficient to consider the set of pairs that agree on `LastName` and `ZipCode` in the address. Note that there is an implicit assumption that comparisons not made due to blocking are non-match records. A good field to be chosen for blocking should contain a large number of values that are fairly uniformly distributed, and must have a low probability of inaccuracy errors; specifically, this last property is due to the fact that errors in a field used for blocking can result in failure to bring linkable record pairs together.

When specific conditions hold, further techniques can be applied to optimize record linkage. In the following, we briefly describe the 1-1 matching technique that can be used when it is known that few duplications occur. Then, we describe the bridging file technique that can be used when a third source is available that links the two sources that are going to be matched.

### 1-1 Matching Technique

The basic idea of the 1-1 matching technique is to force each record of the set `A` to be matched with at most one record of the set `B`. The rationale behind this technique is that if there are few duplicates, it is sufficient to stop to the *best* matching record, which is the record having the highest agreement weight with the observed one. In [105] a technique to force 1-1 matching is proposed, in which the set of matching assignments is globally optimized.

### Bridging File

Given the two files `A` and `B`, the bridging file includes a set of common identifying information for them. For instance, let us suppose that both `A` and `B` store personal information of citizens, namely, `Name`, `Surname`, and `Address`, but `A` stores, in addition, tax-related information, while `B` stores social service-related information. The information common to `A` and `B`, can be available in a *bridging file*, as represented in Figure 5.13. Notice that a record in `A` can be linked to several records in `B`, but typically *not to all*; therefore, the idea is that when a bridging file is available, record linkage efficiency can be improved. However, it is very important to have high quality bridging files, in order to have good matching results.

| A | A&B | B |
|---|---|---|
| $Tax_{1,1}$ | $Name_1$, $Surname_1$, $Address_1$ | $SocialService_{2,1}$ |
| $Tax_{1,2}$ | $Name_2$, $Surname_2$, $Address_2$ | $SocialService_{2,2}$ |
| ... | ... | ... |
| ... | ... | ... |
| $Tax_{1,n}$ | $Name_n$, $Surname_n$, $Address_n$ | $SocialService_{2,n}$ |

**Fig. 5.13.** Bridging file example

## 5.8 Knowledge-Based Techniques

In this section, we describe the details of two techniques that are classified as knowledge-based. Specifically, Section 5.8.1 describes the Intelliclean system and Section 5.8.2 describes the Atlas system.

### 5.8.1 A Rule-Based Approach: Intelliclean

The main idea of Intelliclean [124] is to exploit rules as an evolution of previously proposed distance functions; rules are extracted from domain knowledge and fed into an expert system engine, making use of an efficient method for comparing a large collection of rules to a large collection of objects. Rules are of two types, with different goals:

- *duplicate identification rules*, specifying conditions according to which two tuples can be classified as duplicates. Duplicate identification rules include text similarity functions, but go further, allowing more complex logic expressions for determining tuple equivalence. An example of a duplicate identification rule is shown in Figure 5.14, where duplicates are searched for in a `Restaurant` relation, with attributes `Id`, `Address`, and `Telephone`. For the rule in Figure 5.14 to be activated, the corresponding telephone numbers must match, and one of the identifiers must be a substring of the other; furthermore, the addresses must also be very similar (similarity of addresses using the *FieldSimilarity* function must be higher than 0.8). Records classified as duplicates by this rule have a certainty factor of 80 %. A *certainty factor* (CF) represents expert confidence in the effectiveness of the rule in discovering duplicates, where `0 < CF < 1`. Specifically, we can assign a high certainty factor to a rule if we are sure that it will identify true duplicates. Analogously, we assign smaller values for rules that are less strict.
- *merge/purge rules*, specifying how duplicate records are to be handled. An example is "Only the tuple with the least number of empty fields is to be kept in a group of duplicate tuples, and the rest are to be deleted."

```
Define rule Restaurant_Rule
Input tuples: R1, R2
IF (R1.telephone = R2.telephone)
AND (ANY_SUBSTRING (R1.ID, R2.ID) = TRUE)
AND (FIELDSIMILARITY (R1.address = R2.address) > 0.8)
THEN
DUPLICATES (R1,R2) CERTAINTY = 0.8
```

**Fig. 5.14.** An example of the duplicate identification rule in Intelliclean

The complete Intelliclean strategy is shown in Figure 5.15. The procedure can be seen as an improvement over the sorted neighborhood method presented in Section 5.7.1, where the improvement mainly regards the adoption of rules and a more effective transitive closure strategy.

From step 2.1 of Figure 5.15, we observe that rules are extracted from domain knowledge by domain experts; therefore the approach can be classified as domain dependent. The selection of precise, expressive, and efficient rules is a crucial activity to achieve effectiveness of the cleaning process, i.e., maximize recall and precision (see Section 5.9). Step 2.3 is motivated by the fact

**1. Preprocessing**
Perform data type checks and format standardization
**2. Processing**
**2.1** The compared records are fed into an expert system engine together with a set of rules of the form IF <condition> THEN <action>.
**2.2** Check iteratively within a sliding window first Duplicate Identification rules and then Merge Purge rules using a basic production system to see which ones should fire based on the facts in the database, looping back to the first rule when it has finished.
**2.3** Perform transitive closure under uncertainty using an improved version of the multi-pass Sorted Neighborhood searching method
**3. Human verification and validation stage**
Human intervention to manipulate the duplicate record groups for which merge/purge rules are not defined

**Fig. 5.15.** The complete Intelliclean strategy

that transitive closure in the multi-pass sorted neighborhood algorithm tends to increase false matches. As we have seen in the example, in Intelliclean a *certainty factor* (CF) is applied to each duplicate identification rule. During the computation of the transitive closure, we compare the resulting certainty factor of the merged group to a user-defined threshold. This threshold represents how tight or confident we want the merges to be. Any merges that result in a certainty factor less than the threshold will not be executed.

As an example, let us assume we perform Step 2.3 on the following pairs of tuples: `(A,B)` with `CF = 0.9`; `(B,C)` with `CF = 0.85`; `(C,D)` with `CF = 0.8`; threshold = 0.5. The groups `(A,B)` and `(B,C)` will be firstly considered, as these groups have higher CFs. They will be merged to form `(A,B,C)` with `CF = 0.9 x 0.85 = 0.765`. Then, this group is merged with `(C,D)` to form `(A,B,C,D)` with `CF = 0.765 x 0.8 = 0.612`, still greater than the threshold; however, if the threshold were set at 0.7, `(A,B,C)` and `(C,D)` would remain separate, as the resulting `CF` of the merged group, equal to 0.612, would be less than the threshold.

### 5.8.2 Learning Methods for Decision Rules: Atlas

In Intelliclean, discussed in the previous section, rules are extracted from the domain knowledge by experts, and no specific learning process is conceived for their generation. In this section, we discuss Atlas, a technique, presented in [189], that improves the knowledge-based approach in the following directions:

1. The rules include a wide set of domain-independent transformations, as possible mappings between textual strings, such as `<World Health Organization, WHO>` which transforms a string of three items into the string made of the initials of the items. Examples of transformations are shown in Figure 5.16. `<World Health Organization, WHO>` is an example of the *Acronym* transformation.
2. Structural information on rules can be obtained first from an analysis performed on tuples in the input, in order to extract knowledge on recur-

rent similarities between the different pairs of attributes of objects to be matched.
3. Rules can be obtained through a learning process on a training set, with or without active expert user involvement.

**Soundex** converts an item into a Soundex code. Items that sound similar have the same code

**Abbreviation** replaces an item with corresponding abbreviation (e.g., third → 3rd )

**Equality** compares two items to determine if each item contains the same characters in the same order

**Initial** computes if one item is equal to the first character of the other.

**Prefix** computes if one item is equal to a continuous subset of the other starting at the first character

**Suffix** computes if one item is equal to a continuous subset of the other starting at the last character

**Abbreviation** computes if one item is equal to a subset of the other (e.g., Blvd, Boulevard)

**Acronym** computes if all characters of one item string are initial letters of all items from the other string

**Fig. 5.16.** Examples of transformations

In order to explain in more detail the overall strategy of Atlas, consider the pair of relations shown in Figure 5.17.

Relation1

| Last Name | Address | City | Region | Telephone |
|---|---|---|---|---|
| Ngyo | Mombsa Boulevard | Mutu | MM | 350-15865 |

Relation2

| Last Name | Address | Region | Telephone |
|---|---|---|---|
| Ngoy | Mombasa Blvd. | Masai Mara | 350-750123 |

**Fig. 5.17.** Two relations

In the figure, the two relations have four attributes in common, `LastName`, `Address`, `Region`, and `Telephone`. We assume that the two tuples refer to the same real-world object. The items in the two tuples have several differences, whose nature depends on the attribute. More specifically,

1. values of `LastName` differ, probably due to typing errors;
2. values of `Address` differ, both for a character in the first item and for "distance abbreviation transformation" in the second item;
3. values of `Region` differ in distance "acronym transformation"; and
4. values of `Telephone` match only in the area code, probably due to a different currency.

The four attributes show different behaviors with respect to the differences appearing in the corresponding items. In order to precompute candidate mappings between tuples *similarity scores* are computed for each couple of fields of tuples. They measure

1. local distances between each pair of attributes, based on a composition of applications of transformation and edit distance, applying the cosine similarity measure (see Section 5.4.3);
2. a global distance, where different weights are assigned to attributes in local distances; weights measure the selectivity of the attribute, to reflect the idea that we are more likely to believe matching between attributes in which values are rarer (for definitions and formulas see [189]).

At this point, mapping rules have to be constructed. An example of mapping rule, based on Figure 5.17, is

`If Address` $>$ `threshold1` $\wedge$ `Street` $>$ `treshold2 Then matching`

The *mapping rule learner* determines which attributes or combinations of attributes are most effective for mapping objects, with the final goal of determining the most accurate mapping rules, given threshold values. *Accuracy of mapping rules* is seen as their ability in dividing a given set of training examples in matched/not matched. This is performed by two methods:

1. *Decision trees* is an inductive learning technique, where attributes (and thresholds) are tested one at a time in the tree to discriminate between matching and non-matching pairs of tuples. Once an "optimal" decision tree is created, it is converted into the corresponding mapping rule. In general, this method requires a large number of training examples.
2. An *active learning procedure*, where a committee of decision tree learners that vote is created in order to choose the most informative examples for the user to classify as matching or non-matching.

Once mapping rules are chosen, they are applied to candidate mappings to determine the set of mapped objects.

## 5.9 Comparison of Techniques

In Section 5.3, *search space reduction*, *choice of comparison function* and *use of decision model* were identified as relevant steps in the object identification process. In this section, we first introduce metrics used to evaluate specific steps of object identification techniques (Section 5.9.1). Then, we describe a detailed comparison on two sets of techniques: (i) techniques that are mainly concerned with efficiency issues, i.e. search space reduction methods (Section 5.9.2) and comparison functions (Section 5.9.3); and (ii) techniques that are mainly focused on effectiveness, i.e. decision methods (Section 5.9.4). Finally, in Section 5.9.5, we comment on some experimental results.

### 5.9.1 Metrics

The decision on actual matching `M` or non-matching `U` of two records can give rise to two types of errors, *false positives* `FPs` (also called *false matches* in the

chapter) for records declared as M while actually being U, and *false negatives* FNs (*false non-matches*) for records declared as U while actually being M. *True positives* TPs (*true matches*) are the correctly identified as M and *true negatives* TN (*true non-matches*) are the correctly identified U. Figure 5.18 summarizes such different cases. It follows from definitions that the following equalities hold:

$$\texttt{M} = \texttt{TP} + \texttt{FN}$$

$$\texttt{U} = \texttt{TN} + \texttt{FP}$$

| | |
|---|---|
| M | Actual match w.r.t. real world |
| U | Actual non match w.r.t. real world |
| FP | Declared match while actual non match |
| FN | Declared non-match while actual match |
| TP | Declared match while actual match |
| TN | Declared non match while actual non match |

**Fig. 5.18.** Notation on matching decision cases

Several metrics to evaluate effectiveness of object identification techniques have been proposed, combining such criteria. The most typical metrics are *recall* and *precision*. *Recall* measures how many true positives are identified in relation to the total number of actual matches. It is given by:

$$recall = \frac{\texttt{TP}}{\texttt{M}} = \frac{\texttt{TP}}{\texttt{TP} + \texttt{FN}}$$

The aim of an object identification technique is of course to have a high recall. *Precision* measures how many true matches are identified in relation to the total number of declared matches, including erroneous ones (i.e., FPs):

$$precision = \frac{\texttt{TP}}{\texttt{TP} + \texttt{FP}}$$

The aim is to have a high precision. Recall and precision are often conflicting goals in the sense that if one wants to have a greater number of true positives (i.e., to increase recall level), usually more false positives are also found (i.e., precision decreases). Besides recall and precision, other metrics that have been used are *false negative percentage* and *false positive percentage*. False negative percentage considers how many undetected matches are present relative to the number of actual matches:

$$false\ negative\ percentage = \frac{\texttt{FN}}{\texttt{M}} = \frac{\texttt{FN}}{\texttt{TP} + \texttt{FN}}$$

False positive percentage considers how many wrongly detected matches are present, relative to the number of actual matches:

$$false\ positive\ percentage = \frac{\texttt{FP}}{\texttt{M}} = \frac{\texttt{FP}}{\texttt{TP + FN}}$$

In order to combine recall and precision, *F-score* has also been proposed. It corresponds to the harmonic mean of recall and precision. More specifically, F-score is given by:

$$F - score = \frac{\texttt{2RP}}{\texttt{P + R}}.$$

Besides these specific metrics, traditional time complexity metrics are used to evaluate the efficiency of the object identification process; an example is the *number of comparisons* to be performed during the process.

### 5.9.2 Search Space Reduction Methods

As already described, given two sets of records A and B we want to compare for identifying the same objects belonging to both of them, the search space is the cartesian product $\texttt{A} \times \texttt{B}$. In order to reduce such space, we have seen that three principal methods exist, blocking, sorted neighborhood, and pruning.

Typically, pruning is used in most empirical techniques, either in conjunction with blocking or in conjunction with sorted neighborhood; in the following we will examine blocking and sorted neighborhood. In [65], a comparison of blocking and sorted neighborhood is reported. The two methods are compared considering (i) the blocking method for different values of the block key length and (ii) the sorted neighborhood method for different values of the window size. Blocking and sorted neighborhood are evaluated on the basis of the effectiveness of the matching process, measured by the F-score metric. The experiments show that the F-score values for blocking and sorted neighborhood are comparable for appropriate choices of the blocking key length and the window size.

Furthermore, when comparing the time complexity of the two methods, a comparable behavior is similarly exhibited. Indeed, as already shown in Section 5.4.2, the total time complexity of blocking is $\mathrm{O}(h(n) + n^2/b)$, where $h(n) = n\log n$ if blocking is implemented using sorting, which is comparable to the total time complexity of the sorted method, that is, $\mathrm{O}(n\log n + wn)$.

### 5.9.3 Comparison Functions

Various empirical analyses have been done to discover which comparison functions perform better. In [65] a comparison is reported between 3-grams, bigrams, edit distance, and Jaro algorithm. The experiment considers the behavior of the functions on a set of name pairs, some of which are the same names, but misspelled, while others are different, or swapped. The result of the experiment is that Jaro outperforms for the same name misspelled and known to be different, while bi-gram outperforms for names swapped. In [215], Jaro is again compared with edit distance and bi-gram, and it is shown that it is superior, especially when transpositions are present.

### 5.9.4 Decision Methods

We now characterize the decision methods adopted by the object identification techniques described in this chapter. For each decision method, we report

- *input parameters*, required by the method. Note that some techniques also provide methods for computing such parameters;
- *output*, provided by the method;
- *objective*, that summarizes the main goal to be achieved by the decision method;
- *human interaction*, representing the steps of the object identification process that require an interaction with an expert;
- *selection/construction of a representative* for the matching records, showing which methods explicitly include the selection or construction of a record that represents a specific cluster obtained in the matching process.

The techniques are represented in Figure 5.19.

| Technique | Input | Output | Objective | Human interaction | Selection/Construction of a representative for the matching records |
|---|---|---|---|---|---|
| Fellegi&Sunter | $\gamma$ vector of comparison functions<br>Estimation of $T\mu$ and $T\lambda$<br>m- and u-probabilities | For each record pair, decision on match, non-match, possible match with given error rates | Low error rates (false match and false non-match)<br>Minimization of possible matches | Clerical Review of possible matches | No |
| Cost Based | Matrix of costs of decision rules<br>m- and u- probabilities | For each record pair, decision on match, non-match, possible match with given error rates | Minimization of cost of errors (false match and false non-match) | Clerical Review of possible matches<br>Matrix of costs of decision rules | No |
| SNM | Declarative rules encoding domain knowledge (for tuple level decision)<br>Comparison functions (for attribute value decision)<br>Threshold (for attribute value decision) | For each record pair, decision on match or non-match | Precision/Recall tradeoff | Choice of the matching key<br>Threshold Specification<br>Decision Rules | No (only for incremental SNM) |
| Priority-Queue | Smith Waterman comparison function<br>Threshold (for tuple value decision) | For each record pair, decision on match or non-match | Precision/Recall tradeoff | Threshold Specification | No |
| Delphi | Textual Comparison Function<br>Co-occurrence metric<br>Set of thresholds (dynamically updated) | For each record pair, decision on match or non-match | Precision/Recall tradeoff | None | No |
| DogMatix | XML Threshold similarity (object level) | For each XML element pair, decision on match or non-match | Precision/Recall tradeoff | Selection of candidates<br>Threshold Specification | No |
| IntelliClean | Duplicate Identification Rules (for tuple decision)<br>Merge Purge Rules (for tuple decision)<br>Set of thresholds (for attribute comparison and for tuple merging) | For each record pair, decision on match or non-match<br>Merged Result for matching records | Precision/Recall tradeoff<br>User controlled confidentiality for merging | Duplicate Identification,Merge/Purge Rules Specification<br>Threshold Specification<br>Human verification for merging duplicates when rules are not specified | Yes |
| Atlas | Learnt Decision rules<br>Set of domain independent transformations<br>Thresholds | For each record pair, decision on match or non-match | Precision/Recall tradeoff | Mapping rule learning | No |

**Fig. 5.19.** Comparison of decision methods

Looking at the input column, the decision rules that are used by the method can be specified at attribute and at tuple levels for structured data types. For the techniques that consider relational hierarchies, such as Delphi or XML documents, e.g. DogmatiX, thresholds are specified according to the various elements of the adopted data model. Specifically, in Delphi, thresholds are specified by the comparison between tuples and their children sets; in DogmatiX, the objects to be compared need to be explicitly identified in the XML documents, and thresholds are defined for such objects.

In the output column, observe that the probabilistic techniques typically partition records into three sets, match, non-match, and possible match, at given error rates. Conversely, both the empirical and knowledge-based techniques are used to partition records into two sets, match and non-match. The underlying assumption of such techniques is that of completely automated decision methods, not requiring any human review on possible matches (consider also the human interaction column).

The objective column summarizes the objective of the decision method. The probabilistic techniques rely on formal models explicitly including such an objective. The Fellegi and Sunter model is formulated to minimize possible matches, while the cost-based model has the objective of minimizing the cost of errors. The empirical and knowledge-based methods instead are all validated against the precision/recall performance, namely, how effective the decision method is in detecting true positives (precision) and not detecting false positives (recall).

In the human interaction column, for all methods but Delphi, there is the need of human-defined thresholds. Indeed, Delphi introduces a technique to dynamically determine thresholds, based on standard outlier detection methods, and considering that a duplicate has an outlier-like behavior referred to given similarity metrics.

The representative of a cluster of matched records is actually constructed/selected only by Intelliclean. The concept of cluster representative is proposed also within the sorted neighborhood method and the priority-queue method, but with a different scope, reducing the number of pairwise comparisons to detect duplicates. In contrast, Intelliclean identifies a strategy and appropriate rules for building cluster representatives.

### 5.9.5 Results

The table in Figure 5.20 describes the results obtained by the different decision methodologies and the features of the data sets used for the experiments. For each technique, we report the metrics addressed, the type of data used in the experiments (synthetic vs. real), and the provided results in terms of the different metrics, as claimed by the authors of each technique.

The first row of Figure 5.20 refers to the sorted neighborhood method. Results of experiments are reported for both the synthetic and the experimental data sets. Note that such results depend on a specific parameter, namely, the

| Technique | Metrics | Synthetic/ Real Data | Results |
|---|---|---|---|
| SNM | Precision<br>False Positive Percentage | Synthetic | Precision 50%-70% on independent pass<br>Precision close to 90% with transitive closure<br>False Positive Percentage not significant (0.05 - 0.2%) |
| | Precision<br>False positive Percentage<br>False negative Percentage | Real | Not significant False Negatives Percentage<br>Not significant False Positive Percentage |
| Priority-Queue | Precision<br>Efficiency (Number of comparisons) | Synthetic | Precision similar to SNM<br>Efficiency : 5 times less than SNM |
| | Efficiency (Number of comparisons) | Real | Precision not provided as for real data difficult to identify actual duplicate s<br>EffiNumber of reduced comparisons similar to the one for the synthetic data set |
| Delphi | False Positive Percentage<br>False Negative Percentage | Real | False Positive Percentage less than 25%<br>False Negative Percentage around 20% |
| DogmatiX | Precision<br>Recall | Real | For similarity measure:<br>Experiment 1: Precision 70-100%<br>Experiment 1: Recall: 2%-35%<br>Experiment 2: Precision 60-100% |
| IntelliClean | Precision | Real | Experiment 1: Precision 80%<br>Experiment 1: Less than 8% Recall<br>Experiment 2 :Precision: 100%<br>Experiment 2 :Recall:100% |
| Atlas | Precision (accuracy) | Real | Experiment 1: Precision 100%<br>Experiment 2: Precision 99% |

**Fig. 5.20.** Metrics used by to evaluate object identification by empirical techniques and related results.

size of the sliding window: intervals of values shown in the figure correspond to different sizes of the window. For the priority queue algorithm, the result of an efficiency test is shown, measured by the number of comparisons that the algorithm performs. The results for Delphi concern the first level of the hierarchy (see Section 5.7.3). For DogmatiX the reported results concern primarily the similarity measure included in the approach. The intervals of the metrics refer to the variability of the threshold used for the measure.

The experimental data sets, as well as the experimental conditions and assumptions, are different, and therefore it is not possible to actually compare the different techniques. Nevertheless, the figure's utility is in its summarizing the features of the experimental validation and testing performed on each technique.

## 5.10 Summary

In this chapter we have described several techniques proposed for the most relevant data quality activity, object identification. Due to heterogeneous schemas, and to possible errors in data entry and update processes, objects happen to have different representations and values in distinct databases. As a consequence, a *loss* of a clear identity may affect objects, thus compromising the possibility of reconstructing information sparse in distinct sources. Object identification techniques aim at repairing this loss of identity using

context information available on the similarity of objects' representations in terms of tuples, hierarchical relations, and XML files. The concepts of "context information available" and "similarity" are formalized in different ways in probabilistic, empirical, and knowledge-based techniques. Moreover, techniques proposed in the three areas can be differently characterized with respect to the level of adoption, their efficiency, and their effectiveness. The probabilistic techniques emerge as the most adopted ones, due to their relative maturity and the experiences gained from their application. The empirical techniques have the efficiency as a major objective, and thus are particularly suitable for time critical applications. The knowledge-based techniques have the best potential effectiveness, due to the explicit modeling of domain knowledge. Comparisons between techniques, described in Section 5.9, as well as criteria adopted by specific techniques, provide the reader elements for choosing the most effective technique according to the context. We will discuss these issues in more depth in Chapter 7.