DEGREE OF DOCTOR OF PHILOSOPHY IN
COMPUTER ENGINEERING AND SCIENCE

INTERNATIONAL DOCTORATE SCHOOL IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXVI Cycle

UNIVERSITY OF MODENA AND REGGIO EMILIA
DEPARTMENT OF ENGINEERING "ENZO FERRARI"

Ph.D. DISSERTATION

Heterogeneous Data Warehouse Analysis and Dimensional Integration

Candidate:
Marius Octavian OLARU

Advisor:
Prof. Maurizio VINCINI

Co-Advisor:
Prof. Sonia BERGAMASCHI

The Director of the School:
Prof. Giorgio Matteo VITETTA

DOTTORATO DI RICERCA IN
COMPUTER ENGINEERING AND SCIENCE

SCUOLA INTERNAZIONALE DI DOTTORATO IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXVI Ciclo

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA
DIPARTIMENTO DI INGEGNERIA "ENZO FERRARI"

TESI PER IL CONSEGUIMENTO DEL TITOLO DI DOTTORE DI RICERCA

Analisi di Data Warehouse Eterogenei e Integrazione Dimensionale

Tesi di:
Marius Octavian OLARU

Relatore:
Prof. Maurizio VINCINI

Co-Relatore:
Prof. Sonia BERGAMASCHI

Il direttore della scuola:
Prof. Giorgio Matteo VITETTA

Keywords:

Data Warehousing
Schema Matching
Multidimensional Information Integration
Data Quality

Abstract

The Data Warehouse (DW) is the main Business Intelligence instrument for the analysis of large banks of operational data and for extracting strategic information in support of the decision making process. It is usually focused on a specific area of an organization.

Data Warehouse integration is the process of combining multidimensional information from two or more *heterogeneous* DWs, and to present users an unified global overview of the combined strategic information from the DWs. The problem is becoming more and more frequent as the dynamic economic context sees many companies merges/acquisitions and the formation of new business networks, like *co-opetition*, where managers need to analyze all the involved parties and to be able to take strategic decisions concerning all the participants.

The contribution of the thesis is to analyze heterogeneous DW environments and to present a dimension integration methodology that allows users to combine, access and query data from heterogeneous multidimensional sources. The integration methodology relies on graph theory and the Combined WordSense Disambiguation technique for generating semantic mappings between multidimensional schemas. Subsequently, schema heterogeneity is analyzed and handled, and compatible dimensions are uniformed by importing dimension categories from one dimension to another. This allows users from different sources to have the same overview of the local data, and increases local schema compatibility for drill-across queries. The dimensional attributes are populated with instance value by using a *chase* algorithm variant based on the *RELEVANT* clustering approach.

Finally, several quality properties are discussed and analyzed. Dimension homogeneity/heterogeneity is presented from the integration perspective; also the thesis presents the theoretical fundamentals under which mapping quality properties (like coherency, soundness and consistency) are preserved. Furthermore, the integration methodology will be analyzed when *slowly changing dimensions* are encountered.

Sommario

Il Data Warehouse (DW) è lo strumento principale di Business Intelligence per l'analisi di grandi molli di dati con lo scopo di estrare informazioni strategiche come supporto al processo decisionale.

L'integrazione di Data Warehouse è il processo di unire informazioni multidimensionali da due o più DW *eterogenei*, e di presentare agli utenti una vista globale e unificata dei dati strategici combinati dei vari DW. Il problema sta' diventando sempre più frequente con il contesto economico attuale che vede molte fusioni/acquisizioni di compagnie e la formazioni di nuove tipologie di reti di aziende, come le reti di *co-opetition*, dove i manager devono analizzare tutte le parti coinvolte a prendere decisioni strategiche che riguardano tutti i partecipanti.

Il contributo della tesi è quello di analizzare ambienti di DW eterogenei e di presentare una metodologia di integrazione delle dimensioni che permette agli utenti di unire, accedere e interrogare dati da sorgenti multidimensionali eterogenei. La metodologia di integrazione si basa sulla teoria dei grafi e sulla tecnica di disambiguazione *Combined WordSense Disambiguation* (CWSD) per generare equivalenze semantiche tra schemi multidimensionali. In seguito, l'eterogeneità degli schemi è analizzata e gestita, e dimensioni compatibili sono uniformate attraverso l'importazione di attributi dimensionali da una dimensione all'altra. Questo permette agli utilizzatori di sorgenti distinte di avere la stessa visione dei dati locali, e aumenta la compatibilità dei schemi locali per query di tipo *drill-across*. Gli attributi dimensionali sono popolati con valori d'istanza attraverso una versione dell'algoritmo *chase* basato sull'approccio di clustering *RELEVANT*.

Infine, alcune proprietà di qualità sono considerate e analizzate. Viene presentata l'omogeneità/eterogeneità delle dimensioni dalla prospettiva di integrazione delle dimensioni; in più la tesi presenta le basi teoriche sotto cui proprietà di qualità dei mapping (come correttezza, coerenza e completezza) sono mantenute. Inoltre, la metodologia di integrazione verrà analizzata in presenza di dimensioni di tipo *slowly changing*.

Acknowledgments

The work presented in this thesis would not have been possible without the precious and continuous help and support of those around me.

First of all I would like to sincerely thank my advisors, Prof. Maurizio Vincini and Prof. Sonia Bergamaschi who have always been a helping guide during my studies and my research activity.

For my family I can find no words to say how much I appreciate and cherish their unconditional support in every moment of my life.

Thank you to all my friends who have been there for me.

Thank you, Ema, for making everything possible...

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

The Data Warehouse (DW) is the main Business Intelligence instrument for the analysis of large amounts of operational data. It permits the extraction of relevant information for decision making processes usually inside one single organization. In fact, Inmon defines it as a "*subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process*" [Inmon, 2002].

Traditionally, one single Data Mart (a building block of the DW) is focused on a particular aspect or subject area (thus, *subject-oriented)* and is confined to a single department; the union of all the company's Data Marts form the enterprise DW. These single DWs reflect the needs of a specific department and are focused usually on a single activity of the organization (for example, sales, production process, warehouse management, purchase orders, supply chain management, finance, controlling, etc.). If the DWs are independently built, it is difficult tu subsequently integrate them for analysis purposes.

Data Warehouse integration is the process of combining strategic information from two or more heterogeneous DWs with the aim of providing users a unified view of the entire available information. It is not to be confused with data integration inside one single DW, which is a different process that will be discussed in Chapter 2. The main difference is where the integration occurs. In the first case, independent operational data sources are integrated to form an unique data repository on top of which the DW is built, while te second case considers the integration of the already available multidimensional information from two or more independent, heterogeneous DWs.

The process of DW integration can be considered a specific case of *data integration* and can thus benefit from the decades of research and studies that have been presented so far. However, the problem is *context* specific, and although the required steps to DW integration are similar to the ones in data integration, any DW integration methodology must provide specific solutions to

a number of common issues, like correctly identifying data sources, managing and handling *schema* and *instance* inconsistency, and *reconciling* and *fusing* instance data.

1.1 The Perspective of DW Integration

The problem of DW integration, although more and more frequent, has received little attention so far. There is, in fact, a series of scenarios where managers need to combine data and information from one or more DWs in order to obtain a unique overview of different areas of one single enterprise or a network of collaborating enterprises. For example, in large organizations different departments usually develop their separate, heterogeneous DW without any awareness of other departments. It is thus difficult for managers to have a coherent overview of the entire organization and to be able to take strategic decisions concerning all departments. This situation poses a great limit to the decision making process as the impact of any decision is confined to the area or department that the single DW describes; furthermore, not having a global overview of the entire organization may lead to cases where good decisions for one department can negatively impact other departments' activity.

In such cases, the necessity of integration arises after the DW has been build and presents several difficulties due to the inherent heterogeneity of the data and to the different way different groups manage and represent the same information. The issue may be avoided when the integration goal is clearly defined *a priori* of the development phase. For example, Kimbal proposes a design methodology, the *Data Warehouse Bus Architecture* for creating and maintaining common analysis dimensions for all DWs of the group, eliminating thus schema and instance inconsistencies. The bus architecture creates "conformed dimensions" as either identical or strict mathematical subsets of the most granular and detailed dimensions [Kimball and Ross, 2002]. Following this approach, different DWs can be plugged in a global architecture by means of a simple union, thus information can be seamlessly integrated without large difficulties. The DW Bus Architecture is an integration methodology that eliminates schema inconsistency *by design*.

Unfortunately, in most cases the integration necessity arises after the creation of the DW, and developers and analysts lack the vision of creating a cross-organization DW from the beginning.

Integration is also a requirement when companies collaborate, form alliances, or one company acquires another company. On top of collaboration business structure, managers need to access a series of strategic indicators that describe the activity of all the companies that are involved in the collaboration

effort.

One single company's data is usually managed through a set of specialized tools, each with a specific task. Accessing all of it is a difficult task *per se* because of the distinct models that are used to represent each specific dataset. That is why accessing data from more than one company is not a scalable task, as the difficulty of managing the heterogeneity increases exponentially.

The need for DW integration may be analyzed from a series of perspectives, involving both Information Technology (IT) and business dynamics.

During recent years, the economic and industrial context has seen numerous changes that made companies undergo a series of transformations in the way they perceive and apply business strategies. Some of the reasons may lead back to the financial crisis, others may be related to the natural evolution of companies and the business models they propose.

The classical approach to business is characterized by the fierce competition among organizations to access assets and resources in order to produce *added value* that is offered to consumers. For example, one definition of *Economics* is "the study of the efficient allocation of scarce resources among competing users" [Casler, 1992]. This definition implies that companies are not naturally open to collaborations, especially not with similar/identical companies that have a similar activity type and offer the same kind of services, although there are specific cases where collaboration offers an extra *added value* to the involved organizations (e.g., *supply chains*).

Furthermore, large organizations usually have access to a vast arrays of resources and to the latest technological achievements to allow them to better adapt to market changes and to be up to date with the customers' requirements. On the other hand, Small and Medium Enterprises (SMEs) usually struggle to compete with larger organizations whenever there is competition for new markets and business opportunities. One way of reorganizing business is through *collaboration* with similar organizations that are usually competitors in order to mutually support and divide the active process of business reconfiguration by sharing resources, knowledge and costs. A possible scenario could involve a network of SMEs forming a *virtual enterprise* that can potentially access all the resources of the SMEs. The virtual enterprise, together with other types of approaches offer examples of business collaboration.

Although easy to define as a business objective, business collaboration faces some economical, business process and, ultimately, technological issues. In fact, involved companies need to deeply reason about the benefits of such approach and need to be able to adapt their business models to fit efficiently into a collaborative environment. This means clearly defining common business objectives as well as the depth of collaboration effort, by means of identifying the level of integration among parties. Although there is a general intu-

itive notion of what *collaboration* is, it is often confused with *cooperation*. For example, [Camarinha-Matos et al., 2009] defines *collaboration* as a “more demanding (i.e., than cooperation) process in which entities share *information*, resources and responsibility to jointly plan, implement, and evaluate a program of activities to achieve a common goal and therefore jointly generating value", meanwhile *cooperation* is a similar objective that involves communication and information exchange, but for the achievement of a compatible goal.

The type of interaction among parties thus implies a weaker or tighter coupling of the operational, managerial and strategic business processes, which in exchange determines the type and amount of interaction at operational, managerial and business level. Furthermore, after the business objectives and business processes have been defined, companies need to seamlessly integrate their *information systems* to exchange operationl data and/or strategic information whenever the business agreements may require it. One of the limitations of collaborative businesses is the lack of an information infrastructure to support common decision making and knowledge and information exchange. In fact, nowadays companies rely on a various set of heterogeneous software tools to better manage the organization's operational and strategic activities, like information/management software (e.g., Enterprise Resource Planning - ERP), Business Intelligence (BI) tools (e.g., Data warehouse, reporting services, dashboards, etc.), Business Process Management (BPM) software, Content Relationship Management (CRM), Enterprise Content Management (ECM), accounting software, etc.

These software tools are not entirely compatible among each other, so data integration inside one single company is a challenging task; the situation becomes even more complex when more than one company needs to exchange and integrate data, information and knowledge. Furthermore, information has many ways of representation; there are different, sometimes incompatible models (conceptual, logical and physical) for storing and accessing data and information. This means that different tools of the same type (for example, two ERPs developed by different companies) will have different data models, although conceptually they manage the same kind of operational data. Furthermore, the data model and the instance data also varies with the specificity of the individual company. Thus, the task of creating an actual interface through which different software tools communicate may be minute compared to the difficulties of translating information from one model to another, assuming the task is possible.

The goal of integrating data and structured information is rising also with the increasing adoption of specialized tools by SMEs, which initially used IT only as a support for the operational process. Subsequently, SMEs acknowledged the importance of IT in the strategic behavior when seeking greater com-

petitiveness [Blili and Raymond, 1993]. Small and Medium Enterprises provide an interesting setting as they are good knowledge generators, but are poor at knowledge exploitation [Levy et al., 2001]. This means that they rarely capitalize on the data they produce and are not able to transform data and information into knowledge. Cooperation offers SMEs the possibility to access a larger array of resources and knowledge on which to capitalize for developing a deeper business vision and for obtaining competitive advantage. However, the business opportunity depends on the ability of the companies to efficiently exchange and integrate strategic information.

A scenario of collaborating SMEs is thus an interesting setting where automated information integration methodologies can provide benefits from the business' point of view.

A rather new perspective in business networks is *CO-Opetition* [Brandenburger and Nalebuff, 1996], the simultaneous *cooperation* and *competition* between organizations. Although contradictory as a concept, co-opetition allow companies to collaborate, exchange experience, knowledge and resources, and in the same time compete for strategic advantage. Of course, the ratio between competition and cooperation determines the behavior of the participants, and how much they are willing to share.

Interesting characterizations are presented in [Levy et al., 2001, Sedlak and Tumbas, 2006], where the authors make use of the *game theory* to express the different kinds of synergy between co-opeting SMEs. Without any further analysis of the concept from a business point of view, we point out that the success of such a business network also depends on a proper IT infrastructure that allows knowledge sharing. The role of IS for the co-opeting among SMEs has already been acknowledged by the research community [Levy et al., 2001].

The driving forces behind integration tools adoption in SMEs may be seen both as a cause as an effect of the economic process driving companies. First of all, globalization of communication technology is facilitating the formation of SME networks. These inter-organizational networks, which are strategic partnerships or alliances among SME stakeholders, introduce a new organizational form into assumption [Acs and Yeung, 1999]. In fact, the ability of businesses to communicate at various levels of organization has been in some cases a major drawback for organizations that wanted to form alliances, business networks or other kinds of collaboration and/or interaction.

The type of interaction among organizations implies a weaker or tighter coupling of the operational, managerial and business level, which in exchange determines the type and amount of data and information that the IT infrastructure needs to handle. The advent of Internet and modern enterprise networks, and specialized data management tools provide a great opportunity for collaboration due to the increasing number of complex and specialized tools

for managing larger quantities of data.

Secondly, even when IT is not an incentive for collaboration by itself, it is still required for allowing the business network creation. In this case, the adoption of modern IT tools is not a cause, but an effect of collaboration. Without automated and efficient IT tools, business collaboration is inefficient in such way to potentially compromise the advantages of the collaboration itself.

In any case, IT and information integration proves fundamental both for facilitating and allowing business collaboration.

When businesses collaborate, one of the most important requirements is the integration of the enterprise DWs that allows participants to have a global overview of the collaborating businesses. In this context, the purpose of this thesis is to analyze heterogeneous DW environments, and to describe a DW dimension integration methodology for allowing the integration of multidimensional information in easily accessible repositories.

1.2 Motivating Example

Consider the schemas in Figures 1.1 and 1.2 represent two example Data Warehouses of different organizations or Data Marts of two different divisions inside one large organization. Integrating them is a difficult task if they have been independently developed. Usually, developers must face two types of problems: *schema* and *instance* inconsistencies.

First, the conceptual schemas are different. Although belonging to different companies/groups and built for different purposes, the DWs present similar dimensions: the *time* hierarchy, the *geographical* and the *class* hierarchy. Schema inconsistency means that similar/identical information is represented with a different schema structure. In the examples in Figures 1.1 & 1.2 schema inconsistencies can be observed in all three dimensions; however, the presence

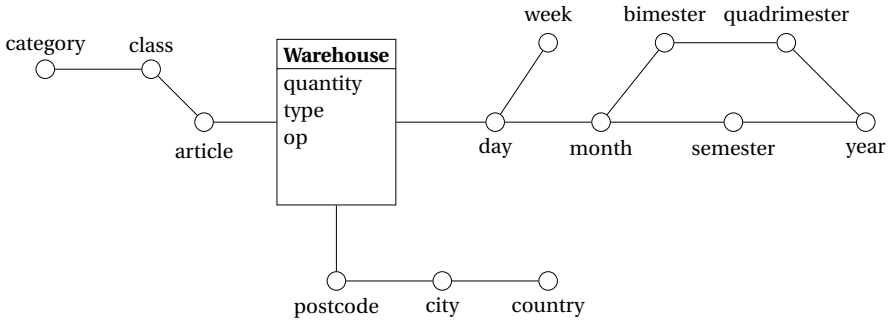


Figure 1.1: Example DW_1

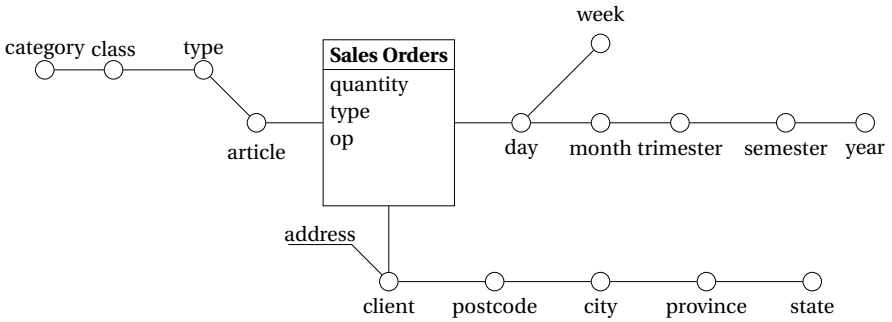


Figure 1.2: Example DW_2

of inconsistencies does not necessarily mean the dimensions are incompatible.

In Figure 1.1, for example, a month is aggregated in bimesters and quadrimesters and finally in years (or alternatively in semesters and years), while in the second DW a month is only aggregated in semesters and years. In this case, the first DW has additional information compared to the second DW. This particular schema inconsistency induces the two different user groups to have a different querying capabilities of the available information, although they have access to the same information.

Apart from a possible different vision over the same information, this particular case highlights a possible integration issue. For example, a user of DW_1 accessing both DWs might be interested in writing the following query: Q_1: *"select the types of articles that have been sold at least 100 times, having stock quantity less than 150, grouped by quadrimester and city"*. The Data Warehouse DW_1 is easily accessible for obtaining stock information (as the time dimensions contain the two required aggregation levels); however, DW_2 cannot provide this kind of information, as facts cannot be aggregated by *bimester*. This means that Q_1 obtains no answer from the two DWs.

Query compatibility presents a serious difficulty when accessing heterogeneous data sources. Whenever a user writes a query over multiple DW, based on the compatibility of the schemas the query can be executed on all the data sources or on a subset of the data sources. In such cases, incompatibilities may be solved in two ways:

1. Integrate information only from compatible sources; this setting is complicated when integrating a large number of data sources as the user loses oversight of which information is integrated; a query integrating information from all the data sources is reliable, meanwhile the opposite case isn't always granted. Furthermore, leaving the user the task of deciding the reliability of the answer of the query only adds complexity and possibly creates frustration.

2. Discard incompatible queries; in this case, the number of valid queries is drastically reduced by the data sources compatibility.

Compared to the previous example, real DWs have a much larger number of dimensions with more aggregation levels. That is why the possibility of encountering schema inconsistencies is much higher. Furthermore, different schemas may also induce different kinds of inconsistencies. For example, a concept may be represented as a dimensional attribute in one DW or a measure in the other DWs. Many of the types of inconsistencies are identical to the ones defined for databases (see [Sheth and Larson, 1990] and [Lenzerini, 2002] for a discussion and examples), although some researchers have defined specific inconsistencies for DW environments (for example [Berger and Schrefl, 2006, Banek et al., 2008]).

Furthermore, in SMEs networks (like co-opetition) developers usually face the need to integrate a larger number of data sources, which further decreases the number of compatible queries.

The second possible inconsistency is the instance level inconsistency, which means representing the same information according to the same conceptual schema, but using different attribute values. Different working groups may represent the same information differently. For example, a month may be represented using the full name (e.g., January, February, etc.) or bay abbreviation (e.g., Jan., Feb., etc.) or by a sequential number (e.g., 1, 2, 3,...). In this case, the information is the same, a direct value-equality approach would not be able to discover that an object refers to the same real-world concept.

Between two distinct, heterogeneous DWs there may be both types of inconsistencies, and information integration tools must be able to correctly discover and handle them. This necessity is twofold: first, relevant information must be identified for integration purposes (*what* to integrate); however, identical concepts must be discovered to allow data reconciliation (*how* to integrate).

Outline of the Thesis
The rest of this thesis is organized as follows:

- *Chapter 2* provides a synthetic analysis of the data integration and schema matching problem. Data integration inside Data Warehousing is distinguished from DW integration, and different conceptual approaches for integration are presented. Furthermore, the chapter contains a brief overview of the related work presented so far in the research literature.

- *Chapter 3* presents an analysis of heterogeneous DW environments, and divides between inter- schema and intra-schema heterogeneity. The implications of intra-schema heterogeneity on summarizability and dependent `GROUP BY` queries is also analyzed.

- *Chapter 4* describes the dimension integration strategy for heterogeneous DW environments. The methodology is divided between schema matching, schema integration and instance integration. An interesting result presented in this chapter is that the integration methodology is independent of the integration architecture and can be successfully used in different integration settings.

- *Chapter 5* presents a quality analysis of the mapping among dimension categories. Three different properties are analyzed, mainly *coherency*, *soundness* and *consistency*. Furthermore, intra-schema heterogeneity is discussed as well as the preservation of homogeneity after the integration step.

- *Chapter 6* presents the conclusions of the current thesis.

Chapter 2

Data Integration and Schema Matching

2.1 Data Integration

The problem of *data integration* has been tackled systematically during the last several decades and continues to be a hot research topic with the proliferation of information systems both inside organizations as for personal use. Formally, “data integration is the problem of combining data residing at different sources, and providing the user with a unified view" [Lenzerini, 2002].

The nature of the problem resided initially in the lack of formalisms for representing data, and subsequently in the lack of standardization in the use of data formalism for representing the same datasets. This later case is called *logical heterogeneity* [Hull, 1997] (or *semantic heterogeneity* [Ceri and Widom, 1993]) and results from the fact that the same information is contained in different, overlapping data repositories and is represented with different formats or different instance values.

A simple example to the reader could be a structured dataset (database relation, electronic spreadsheet, a flat file, etc.) containing data about students of an university. Simple concepts like *name* and *surname* may be represented by two distinct attributes by some developers, while others may represent them in an unique attribute *Name and Surname*. While not being incorrect, the two cases highlight how even simple and well understood concepts can lead to different representations. If we add the fact that the conceptual, logical and physical representation of data and information is obtained after the requirements and specifications pass from users to analysts, data specialists, developers and other figures each having its own level of expertise and ultimately its own vision on data representation, then an infinity of different representations may be ob-

tained for the same conceptual data, all of them potentially correct, all of them more or less accurate and complete.

Many researchers have provided critical discussions on the different kinds of semantic heterogeneity (e.g., [Kim et al., 1995, Hammer and McLeod, 1993, Batini et al., 1986, Kim and Seo, 1991, Kedad and Métais, 1999]) for database systems, although many of them are common to the ones in multi DW environments. In any case, identifying the types of heterogeneity between data sources to be integrated is only the first step in developing an integration solution for heterogeneous data sources.

The difficulties to be tackled during data integration are numerous, also the solutions vary from one application domain to another. Implementing a data integration solution, developers need to handle a series of issues that are relevant to all types of data management systems, including structured and semi-structured data [Hull, 1997]:

1. provide an integrated view of overlapping data sets;

2. identify and specify the relationship between two or more instantiations of replicated data;

3. keep replicated data “synchronized".

Given that the schemas and instances of data sources to be integrated are usually different, besides schema heterogeneity, any integration system must also handle instance inconsistency. This is achieved generally through the use of two components: *wrapper* [Wiederhold, 1992] and *mediator* [Ullman, 1997]. A wrapper is a tool that can access a local data source, extract and translate data from the local schema to an external global schema. An integration system is usually composed by two or more wrappers that feed data to a mediator that reconciles and presents it to the final user or to other systems as an unified view. The unified view is the only point of access for the final users that see the unified view independently of the integrated sources. This provides simplicity to the end user that can better focus on the interrogation and manipulation of the integrated repositories, rather than on the integration process.

The relation between the global schema and local schemas is characterized mainly by two types of approaches: *local as view* (LAV) or *global as view* (GAV) [Lenzerini, 2002]. The first assets that each local schema is a view over the global schema and can be used whenever there is a global reference model for the local repositories. The second considers the global schema as a view over the local schemas [Halevy, 2001]. The global schema is obtained by analyzing, mapping and synthesizing the local schemas into an unique reference

model that is able to simultaneously express the concepts of each independent data sources and to offer users a transparent, unified overview.

A *global-local as view* (GLAV) approach [Friedman et al., 1999] has also been proposed to combine the advantages of the previous approaches. Without analyzing the approaches further, it must be said that the type of considered approach changes the semantics of the querying and integration process.

One other important aspect of data integration is the architecture of the integration framework. Simplifying the numerous types of proposed architectures, one may conclude that the most frequently adopted solutions for data integration are central integration repository, federation approaches and data exchange scenarios for distributed networks.

In the first case, a global point of reference is considered for integrating data from various sources (many such systems have been built, e.g. [Beneventano et al., 2000, Chawathe et al., 1994, Carey et al., 1995, Roth et al., 1996]) either through a materialized view or through a mediator that accesses the individual data sources and extracts it for subsequent reconciliation and integration. A federation approach is similar to the first case, but is focused on maintaining the physical division of the data sources. In distributed networks (like peer-to-peer networks of data sources) data is exchanged between single data sources (some examples [Arenas et al., 2013, Fagin et al., 2005b] for a discussion over the approach). For this later case, a solid formalization has been provided in [Miller et al., 2001], for the CLIO integration approach.

Among the main differences between building a central integration repository and data exchange is that in the first case, (see Figure 2.1 for the architecture) a single point of access is offered to the user that can seamlessly access the combined data sources through the interface. In the second case (see Figure 2.2) data is exchanged directly between two points of interest through a series of mappings and translations rules that converts data from the source

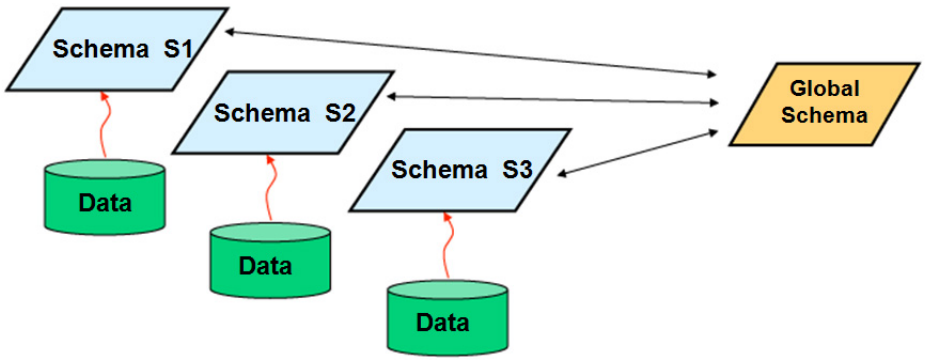


Figure 2.1: Mediator (global schema) architecture

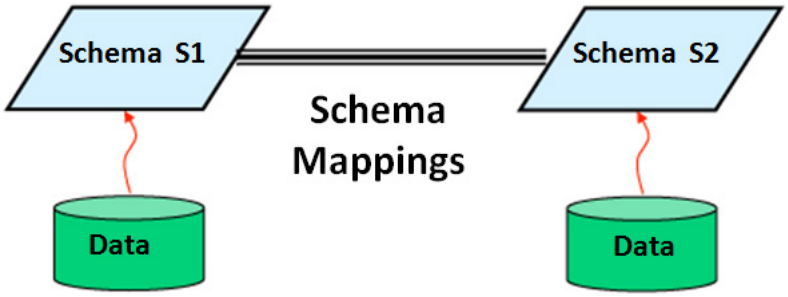


Figure 2.2: Data exchange architecture

schema to the target schema (see [Fagin et al., 2005b, Fagin et al., 2005a] for a critical discussion).

The different kind of integration architecture radically changes the semantics of the integration process. Without extending the discussion to the rigors of a critical analysis, it must be said that the main advantages of a mediator architecture is the unified global view through which the users have the possibility of accessing all the data sources contemporaneously with one single query, while the main disadvantage is the complexity required to analyze and synthesize a global schema, as in real-life cases data integration systems need to combine a multitude of sources. Such systems rely on wrappers that can interface the mediator to each local source either for rewriting the global query into a local query that is compatible to the schema of the local source, and convert the result to the schema required by the mediator. The process introduces an overhead in accessing the data sources, both when rewriting the global query, as when translating and integrating data from all the data sources into an unified answer.

On the other hand, a data exchange architecture provides the flexibility of choosing the sources to be integrated based on selection criteria, assuming that not all of the data sources need to be queried. One possible scenario is a *peer-to-peer* like network of data sources [Bernstein et al., 2002, Halevy et al., 2003, Aberer, 2011] where each member has the possibility of choosing which data sources to query. The downside is that the network is inefficient when accessing a large number of data sources as data and information is obtained through queries that can potentially *flood* the network and uselessly replicate data throughout the network.

2.2 Data Integration in Data Warehousing

Data integration is one of the most important processes in Data Warehousing and has long challenged developers and analysts both from a conceptual point of view (integrate data correctly) as from a technical point of view (regarding mainly efficiency). In fact, a DW usually contains an overwhelming amount of data (both actual as historical) which requires many computational resources to be processed.

Data integration in a DW is not to be confused with DW integration. The next sections better describes the two processes and the different kind of approaches of achieving them.

In [Golfarelli and Rizzi, 2006] a DW is defined as "*a container of integrated historical data ... that allows management to easily extract reliable information in support of the decision process*". As it turns out, companies rely on a complex set of data repositories to handle operational data. The DW has to contain all of them, thus *integration* is one of the most important aspects of the enterprise DW [Calvanese et al., 1999, Calvanese et al., 2001, Calì et al., 2003].

During the integration phase, data passes through a series of complex cleaning [Galhardas et al., 2001] and transformations steps. The main objective is to clean it of inconsistencies, increase its quality and render it easily accessible to the complex set of OLAP tools.

Simple DWs may be obtained by creating views over the relational data (in [Calvanese et al., 2001] a DW is defined as "*a set of materialized views over the operational information sources of an organization*"), however in most of the cases the DW feed process requires the use of complex *Extract-Transform-Load* (ETL) tools (see [Vassiliadis, 2009] for a discussion on ETL methodologies) that can better process the complex elaborations that data undergoes. The ETL phase is often the most laborious and time consuming of the Data Warehousing process [McFadden et al., 1998].

The variable complexity of data transformation inside one single DW gave rise to different types of DW architectures that can better cope with the high volumes of data that is usually loaded in the DW. In [Golfarelli and Rizzi, 2006] the authors summarize the DW architectures into three types, alongside the advantages and disadvantages of each approach.

One level DW architecture. This kind of architecture implements the DW as a multidimensional view over the operational data [Devlin, 1996]. It is the most simple DW architecture and implies that the warehouse level is not materialized, rather it is dynamically created and accessed at query time. A positive note of this approach is the lack of data redundancy, as the multidimensional views are not materialized. One major drawback, however, is the lack of

separation between the transactional and analytical layer. The complexity of the transformations of the data is also limited, since the views are recomputed at query time.

Two levels DW architecture. In order to achieve independence between the operational and analytical layer, a temporary data repository is introduced [Lechtenbörger, 2001]. This allows data to be handled and managed more independently by means of specialized ETL tools [Jarke et al., 2002]. Also, the intermediate data level allows the integration of data from different repositories through the use of ETL tools (see Figure 2.3).

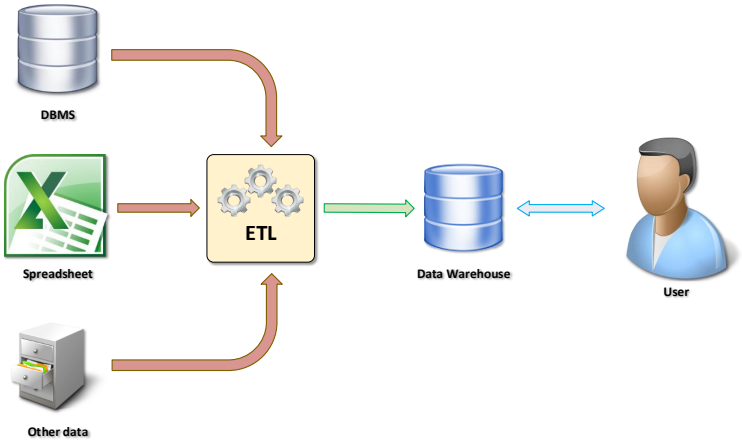


Figure 2.3: Two level DW architecture

Three levels DW architecture. In some architectures, a third level is introduced for *reconciled* data. This is needed for obtaining a materialized view of the enterprise data prior to building the DW. The approach is somewhat similar to a two-levels architecture as data is also integrated from various sources. However, the main difference is that in the two levels architecture, the data reconciliation layer is virtual, meanwhile in the three levels architecture it is materialized.

The advantages of materializing the data reconciliation level is to allow companies to have a unique reference model for the integrated data to be processed by complex transformations before being loaded in the DW. Figure 2.4 illustrates the main elements of a three level DW architecture. Note that in this case the reconciled data layer feeds the DW loading process directly, and may also be a building block for other complex enterprise IT tools.

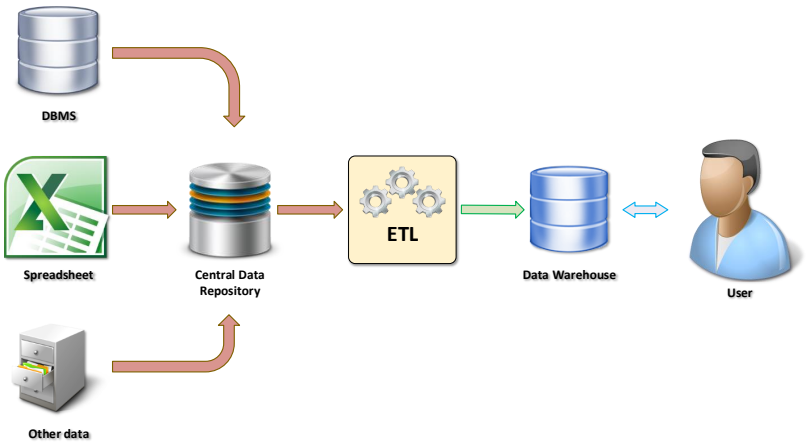


Figure 2.4: Three level DW architecture

Due to the complexity of the data elaborations inside the DW, the two-levels and three-levels architectures are by far the most widely used by practitioners.

The list of the different architecture types for the DW serves for presenting the complex transformations that data undergoes during the load and refresh process. Before being loaded in the first dedicated repository, data is integrated from various sources, a process that requires identifying similar and compatible data sources, analyzing the schema and instance of each data source and generating an integration solution (by choosing a GAV or LAV approach). This first step serves as a basis for subsequently building the enterprise DW. Data then undergoes a series of complex operations with the main objective of increasing its quality (various checks are being made in this step, both at schema and instance level), then it passes through a series of transformations that renders it more easy to query through the designed multidimensional model. Also, data is usually *denormalized*, primarily to increase query performance during DW interrogation. This later transformation adds an extra overhead to the DW loading procedure.

2.3 Data Warehouse Integration

A simplified overview of the architectures can be divided in three important components: data sources, ETL and Warehouse. The *data access* layer is represented by the various data sources, and is considered the *low*-end part of the DW architecture, while the final Warehouse containing multidimensional information is the *high*-end of the DW architecture.

2.3.1 Low-end Data Warehouse Integration

When operating in a multi-enterprise scenario, the classical approach to Data Warehousing (extract data, integrate and reconcile, then build the DW) suggests that in order to build a new, unified DW, all data sources belonging to the organizations must be first integrated, reconciled and unified prior to loading it in the global DW. This is what usually happens during company merges/acquisitions, when analysts usually adapt (transform) data from the new repositories to the model of the DW reference data layer. If the newly acquired company contained its own DW, then this is usually discarded, with the new data flowing into the main DW. The work in [Maurino et al., 2013] describes a real example of this solution.

Although this approach is motivated by the classical approach to Data Warehousing, it has several major inevitable drawbacks. First of all, the complexity of adapting a data source schema to an existing model is not always feasible. The limitations induced by schema incompatibility may drastically reduce the value added by the new repository. Also, although the approach is feasible when adding few new data sources, the complexity is overwhelming when synthesizing a global schema from a large number of different, highly heterogeneous data sources, like when building a DW from repositories of large networks of organizations.

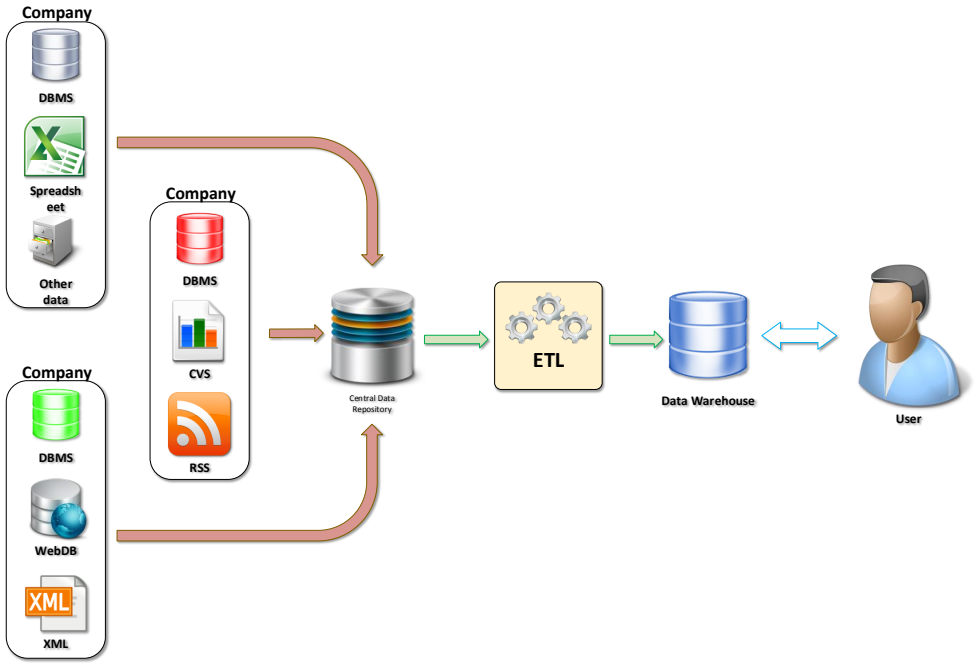


Figure 2.5: Low-end DW integration architecture

Secondly, the precious and complex effort in creating each individual DW is lost, because rather then integrating the already available multidimensional information, the data sources are first integrated in the main repository while the logic under which they are elaborated is subsequently re-developed.

Finally, a batch operation refreshing the large amount of operational data requires high resources.

2.3.2 High-end Data Warehouse Integration

An alternative and innovative approach for heterogeneous DW environments is to integrate the already available multidimensional information contained in the individual DWs through a methodology that is able to recognize and handle the multidimensional model used for representing information in the DW, map similar DW attributes and coherently integrate either by creating a central warehouse that contains all the integrated information or by allowing each individual DWs to seamlessly exchange information whenever the business process may require it.

Such an approach has its advantages and disadvantages, which will be summarized in the following. Among the advantages, the most important are greater data quality, reduced loading time, greater business network flexibility and the reduction of data latency and replication.

Data quality. The integration of more than one data source into an unique, reconciled repository is a laborious task. Achieving the goal inside one single

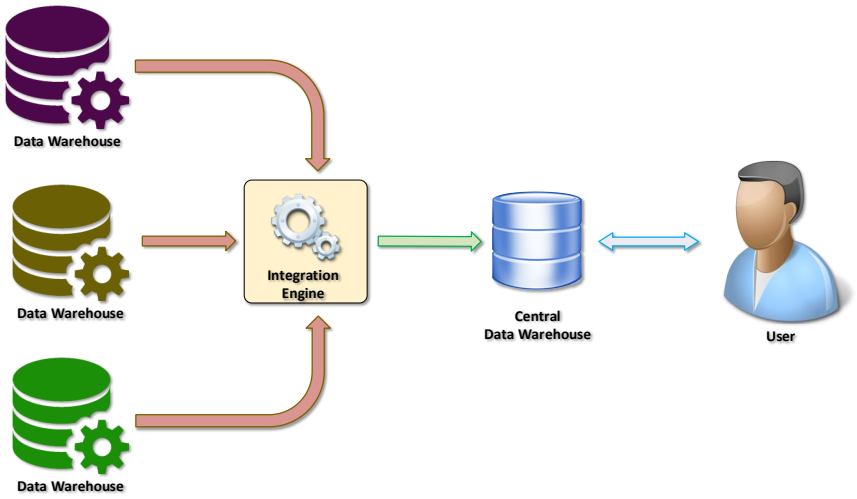


Figure 2.6: High-end DW integration architecture

company, where the data sources, even if different, are related proves to be extremely difficult. Nonetheless, integrating multiple, heterogeneous operational data sources from more than one company means having to deal with an elevated number of schema and instance inconsistencies. In fact, developers find themselves in cases where, due to schema and/or instance incompatibility, have to develop DWs based on partial, incomplete or even misleading information. This situation is useless or risky in real cases, as the level of quality required inside DWs is high. Managers, in fact, need to be able to make decision based on complete and extremely accurate information; otherwise, they may risk damaging the decision making process itself [Ballou and Tayi, 1999].

It is clear how having to cope with the integration of a potential high number of data sources makes the entire process prone to errors. However, integrating the already available high-end, multidimensional information can reduce the complexity of the entire process, greatly reducing the risk of generating importation errors. Moreover, the information contained in each DW has already been cleansed and transformed in order to be easily interrogated, so the process needn't start from the raw operational data, but from the information available in the final DWs.

Time consumption. During the DW building procedure, the ETL development phase is the most time-consuming, in some cases arriving at 60-70% of the entire development process. Developers have to identify the required data sources, to create a conceptual schema of the final information that the business scenario requires (by using a bottom-up, top-down or mixed approach), and then to correctly map the initial operational data to the final information schema by implementing intermediary processing steps through elaborated ETL procedures. Although this phase may be in some cases automated (for example, by using a more advanced ETL, like a semantic ETL [Bergamaschi et al., 2010]), the possibility of integrating the already available multi-dimensional information and skipping most of the already available ETL processes has the advantage of considerably reducing the development time and effort required for the DW building procedure.

Business network flexibility. When using a traditional approach to integrating multiple, heterogeneous DWs, independently of the chosen architecture, the structure itself is static. Imagine, for example, the case in which a set of companies decide to collaborate, and after having identified the required data to integrate among their information repositories, the ETL sequences are developed for the actual integration. As pointed out, the process itself is time consuming and prone to errors. If at the end of the integration process, one or more com-

panies decide to leave or to enter the business network, part or the integration process must be reengineered from the beginning, analyzing the compatibility of the new data schemas with the already available one, and reconciling both the schema and the instances of the given data sources.

In the best of cases, the global schema isn't altered by the discarded or the new data repositories, so the entire operation is achievable by simply discarding the deleted data source or by mapping the newly inserted data source to the global reference model. In the worst of cases, the process implies reengineering parts or the entire global schema, starting from the individual data sources.

Again, having the possibility to integrate the already available multidimensional information allows a more rapid change in the structure of the business network, permitting organizations to more easily enter or leave the network and thus increasing the dynamicity of the network itself.

Data latency and replication. One of the major drawbacks of today's Data Warehousing is the latency of the information gathered in the final DW. In fact, the ETL process is time-consuming not only when implemented, but also when it is executed, due to the extensive transformations that data usually has to undergo. When using a classical approach to DW integration (i.e., by means of ETL procedures), the process must be executed twice, for loading the local DW (that is sometimes maintained) and for refreshing the global DW. In this situation, there is also the issue of the data replicated both in the local DW and in the final DW.

In addition, another important issue is that the time required for the refresh procedure of the DW increases exponentially with the amount of data, mainly due to the massive join queries executed for the retrieval of the required data. That is the reason why splitting the process to the individual DW load process and subsequently integrating the final DW has the potential of reducing the overall required time for refreshing the global DW. Moreover, managers of the individual companies can already access the local DW before the global Warehouse has been refreshed.

Data privacy and security. Data privacy is one of the most important aspects in some specific contexts of DW integration. For example, in *co-opetition* organizations are collaborating and competing at the same time. The type of mutual collaboration and competition among parties determines, among other things, the type and amount of data and information they are willing to share. The reasons is that collaboration can bring mutual benefits for the main participants, however it can also have a *negative reverse-impact* (NRI) [Loebecke et al., 1999] because companies willingly offer private (and potentially sensitive) data to direct competitors. In a *low*-end DW integration architecture, the privacy and

security issues become more stringent as every companies gives direct access to some parts or its entire operational data.

Information inside the DW is mainly derived by aggregations of operational data, as business analysts do not focus on individual events (e.g., a client bought a specific product in a specific day), rather on strategic indicators of event types grouped by the dimensions of analysis (e.g., the total number of elements of a certain type sold per day and per store). It is clear how the DW introduces a certain degree of *anonymity* to operational data, thus increasing the level of privacy. Following this perspective, a *high*-end DW integration proves being a good mechanism for saving data privacy. Furthermore, companies have greater control on which information to share.

Although provides great advantages, high-end DW integration has its potential risks and limitations. First of all, the multidimensional schemas of the repositories to be integrated is more complex, thus from one point of view it is easier to encounter schema heterogeneity, and from the other point of view it is more difficult to discover and correctly map similar concepts, mainly because the semantic similarity of two concepts must also be expressed by means of hierarchical relations[1].

Secondly, a DW may contain only a portion of the information that can be extracted from an operation repository. In this case, integrating the already available DW the users have access only to the information it offers. This is of course a limit when the issue is replicated to most of the DWs.

2.4 Schema Matching

2.4.1 Introduction

A schema is a conceptualization of a domain of interest. It has been used in information technology and database theory for representing data and information that is understandable by computation machines in order to automatically process large quantities of data in only a fraction of the time required by a human operator to handle the same amount of data. In the last decades, many models and formalisms that are now used for for representing most of the structured data have been proposed by different research groups. Each of these models are focused on a specific approach and reflect the requirements of users and applications accessing data in a certain context.

[1] For example, the concepts *day* and *month* are both used to quantify time, but one is more detailed than the other.

To cite only a fraction of the most common, one must refer to *structured* data, where Database Management Systems (DBMSs) have been successfully used for handling and manipulating large quantities of data, and *semi-structured* data that is used where the requirements are different, and more focused on easy data exchange and access. In database theory, analysts and developers distinguish three levels of modeling, the conceptual schema (the *Entity-relationship* [Chen, 1976] being perhaps the most used approach), the logical schema (the *relational* [Codd, 1970] model was and continues to be the *de facto* standard for DBMSs) and the physical layer that is strictly dependent on the hardware architecture and the DBMS technology. Other approaches include the *object-oriented* databases [Atkinson et al., 1992] (first introduced in [Atwood, 1985]), *geographical databases* [Westlake, 1988], etc. Among semi-structured models it is fair to cite the Extensible Markup Language (XML) [Bray and DeRose, 1997, Bray et al., 1997] and other kinds of markup languages.

It is clear at this point how semantic and syntactic heterogeneity are further aggravated by the different models that working groups often use for representing the same information. In this context, the process of *schema matching* plays a crucial role in data integration.

in [Rahm and Bernstein, 2001] schema matching is defined as the process of characterizing the interschema relationships between two distinct schemas. Given the two schemas S_1 and S_2, a schema mapping produces a mapping between elements of the two schemas [Doan et al., 2000, Palopoli et al., 1998, Milo and Zohar, 1998, Miller et al., 1994].

The mapping may be interpreted as a partial or a total function $f : S_1 \rightarrow S_2$, where S_1 and S_2 are structured sets of attributes.

The rapid growth of information has given rise to a series of integration solutions, all of them relying on the schema matching process, that plays a central role in numerous applications, such as web-oriented data integration, electronic commerce, schema integration, schema evolution and migration, application evolution, data warehousing, database design, web site creation and management, and component-based development [Rahm and Bernstein, 2001].

2.4.2 Schema Matching Classification

The dimension integration methodology presented in this thesis is based on a cardinality-based schema-matching technique. To have a better understanding of the main advantages and disadvantages, the current section provides a brief description of the different schema-matching approaches.

Several surveys have been proposed so far by different researchers (for example [Euzenat and Shvaiko, 2007, Bergamaschi et al., 2011, Shvaiko and

Euzenat, 2005]), although this section presents the classification proposed in [Rahm and Bernstein, 2001], where the authors distinguish the approaches based on the following criteria.

Semantic matchers may be either **individual** or **combinational**. The individual matchers use a single criteria for generating the semantic mappings, meanwhile the second approach uses a combination of individual matchers, either by using multiple matching criteria within an integrated *hybrid matcher* or by combining multiple match results produced by different match algorithms within a *composite matcher*. Individual matchers can further be classified using one of the following criteria.

Instance vs schema: matching approaches can consider instance data (i.e., data contents) or only schema-level information. The available information includes the usual properties of schema elements, such as name, description, data type, relationship types (part-of, is-a, etc.), constraints, and schema structure. Schema-based matching algorithms are efficient when naming conventions are standardized and there is a general consensus about how the data should be organized.

Instance-based matching requires that the instances of the schemas to be matched have common features. The main drawback of this approach is that there are several situations where data instances are not available due to security reasons or restricted authorizations [Sorrentino, 2011].

Element vs structure matching: match can be performed for individual schema elements, such as attributes, or for combinations of elements, such as complex schema structures. An element-level matcher determines for each element of the source schema its matching element of the target schema. On the contrary, a structure-level matcher handles combinations of elements in cases where, for example, attribute sets match only partially. Structure-level matching requires that the structures of the schemas to be matched are fully/partially compatible, meaning that they share common parts.

Language vs constraint: a matcher can use a linguistic based approach (e.g., based on names and textual descriptions of schema elements) or a constraint-based approach (e.g., based on keys and relationships). Language-based or linguistic matchers use names and text (i.e., words or sentences) to find semantically similar schema elements. The linguistic matching may be based on the *label* (or name) of the elements or on their description. The text (label or description) is usually stemmed and tokenized, and words are analyzed according to various types of relationship: equality of names, equality

of synonyms, equality of hypernyms, substring similarity or user provided matching criteria.

Constraint-based matching uses the implicit or explicit constraints that structured data contains. For example, equivalence rules can be based on the equivalence of data types and domains, of key characteristics (e.g., unique, primary, foreign), of relationship cardinality, or of is-a relationship.

Matching cardinality: Depending on the schemas and the matching algorithms, equivalences may be produced between individual elements or between groups of elements. This gives way to four different cases of relationships, $1:1$, $1:n$, $n:1$ and $m:n$. In the first case, elements are mapped individually because they are semantically equivalent. In $1:n$ and $n:1$ mappings, the meaning of an element in one schema is equivalent to the information contained by a group of elements in the other schema. For example, a person's name may be represented by the string attribute *Name and Surname* in one schema, while in the other schema it may be stored in two different attributes *Name* and *Surname*. In this case, the cardinality of a correct matching is $1:2$.

Auxiliary information: Additionally, matchers may rely on other information for schema matching, like ontologies, thesauri, user decisions, previous matching decisions, etc.

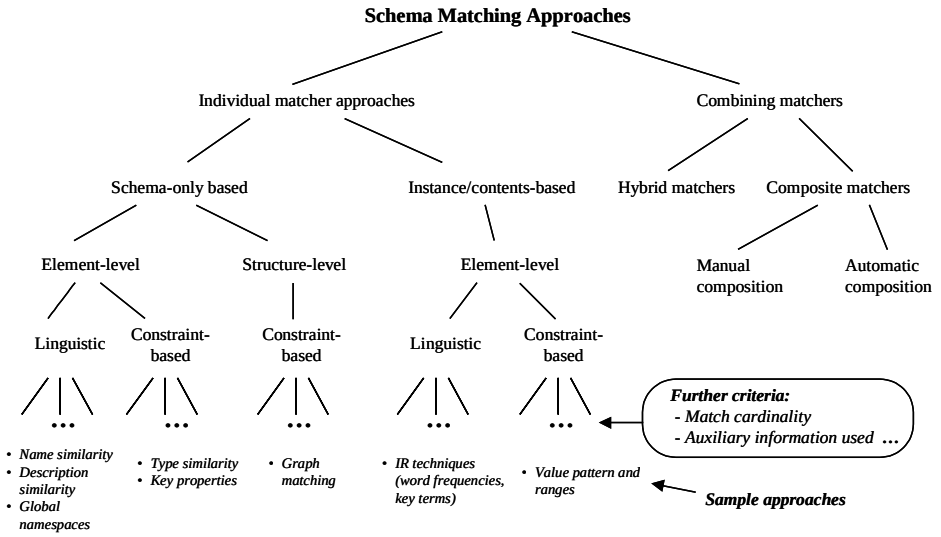


Figure 2.7: Classification of schema matching approaches

Figure 2.7[2] summarizes the various classifications of individual and combinational matching approaches.

By using the previous classification, the integration methodology that will be later described uses both schema as instance data to match single elements of DW dimensions, although an intermediate step considers parts of the schemas for identifying similar elements. The mapping cardinality is $1:1$, as dimension categories are mapped individually. Furthermore the approach is constraint based, although it uses semantics for mappings pruning.

2.5 Related Work

In order to have a better understanding of the contribution of the thesis, the current section provides a literature overview of some of the DW integration and data integration related to Data Warehousing approaches presented so far in the research literature.

As explained in Section 2.3, the integration may be of two types: low-end and high-end DW integration. As such, different DW integration methodologies and solutions have concentrated on these two distinct areas of the DWs to be integrated and provided optimizations in some cases and innovative solutions in other cases. It is fair to say that although high-end integration provides numerous advantages that have been discussed in this chapter, it hasn't always been the implemented solution in real life cases. The reasons are at least two.

First of all, low-end integration can benefit from the decades of data integration and data exchange research and developers can already implement available solutions that have been successfully proposed both by academia and private companies. The maturity of ETL tools (already in use in Data Warehousing) provide a strong incentive for developers to use a solution they are already acquainted with. Furthermore, high-end DW integration has received the research community's attention only recently. Secondly, a high-end solution requires analysts to have a wider vision at a more abstract level of the data they handle and often they prefer to implement well-known solutions that provide lower business risks to the organizations.

The best example of low-end DW integration is provided by [Maurino et al., 2013] where an application to *co-opetitive data warehouse* is proposed. The co-opetitive data warehouse is described as "*a data warehouse where there is the need to a real integration of business data coming from the day-by-day activity provided by organizations exposing a co-opetitive behavior*". The paper aims at analyzing the differences and similarities from traditional Data Warehousing,

[2] The figure was extracted from [Rahm and Bernstein, 2001]

and describe a real case of co-opeting organizations. The integration of the operational data sources in the presented example is based on the combination of *virtual data integration* (through a global virtual view - GVV) and traditional ETL techniques, and the interaction among participants is modeled by means of *game theory*.

Among the main aspects of the implemented methodology, the authors specifically consider *privacy preserving strategies* against data mining techniques [Aggarwal and Yu, 2008] and the impact of *data quality* on the integration architecture. It is interesting to note that in a high-end DW integration the privacy issues are eliminated, as the integrated information is summarized. In this way, there is no need for privacy preserving intervention on the operational data. Data quality is expressed in terms of the heterogeneity of the individual data sources to be integrated. The integrated GVV has to be cleansed and checked prior to being translated to a multidimensional model. As discussed in the current chapter, this operation is not needed in high-end integration as the information contained in the DWs comes from already cleaned and checked operational data sources.

The work in [Tria et al., 2013] describes an ontology-based approach to source data integration in Data Warehousing. The main idea of the paper is to automate the integration process by using a shared reusable ontology built from the logic-based description of the entities forming the individual data sources. The methodology assumes that designers manually build a conceptual schema along with a data dictionary for every data source, storing the description in the natural language of the concepts modeled by the database. The approach is iterative, based on an binary operator that, given two conceptual schemas, produces a new conceptual schema that integrates the concepts contained in the two schemas. The process starts by analyzing and merging the first two schemas S_1 and S_2 into a common global schema G_1; then it iteratively adds the remaining schemas to the global schema, by using the following iteration step: $G_i = integration(G_{i-1}, S_i)$.

Although provides a more refined solution, the integration in this approach is still low-end, and has all the disadvantages deriving from it. Given the multitude of operational data sources that may be present in each DW, the complexity of deriving a common global schema synthesizing all the data sources may render the process unfeasible. Furthermore, the user has to manually provide a schema for each data source, hence the process requires considerable human intervention.

Another more refined approach to *low-end* DW integration is provided in [Bergamaschi et al., 2010], where the authors propose a semantic approach to ETL technologies. The paper describes a tool based on the semantic analysis of two or more schemas for defining mappings among the data sources and the

DW, and a set of clustering techniques for defining transformation functions that manage, reconcile and integrate data from the repositories. The goal of the approach is to automate both the extraction as the transformation process of data. The authors rely on the MOMIS data integration system [Beneventano et al., 2003] and the *RELEVANT* clustering approach [Bergamaschi et al., 2007e].

In [Alqarni and Pardede, 2012], the solution provided by the authors is done at a higher level compared to the previous solution. In fact, the authors make use of a reference multidimensional conceptual level (namely *X-Warehousing* [Boussaid et al., 2006]) to describe the DW and to subsequently incorporate unstructured data from business documents. The unstructured data is assigned a generic XML schema that is mapped to the reference DW XML schema by using the methodology in [Boussaid et al., 2006]. Prior to being loaded in the DW, data is tokenized and stemmed. Two interesting remarks can be made regarding this approach: (1)it considers the DW schema for integration purposes (not the operation sources schemas); (2)it can be used for integrating several DWs by automatically integrating the operational data sources into the reference DW. In fact, although the authors propose their methodology for unstructured business documents, it can also be used for generic DW integration by translating the schema of any operational data source into XML format, and subsequently using the methodology in [Boussaid et al., 2006] to integrate it into a global DW.

The work in [Wu and Hå kansson, 2010] tackles a higher aim, that of DW metadata integration through a Knowledge Based System (KBS). The approach is based on the Common Warehouse MetaModel (CWM) [Object Managenemt Group, 2003] and utilizes the knowledge of engineers on CWM and the metadata models to give metadata interchange model suggestions and thus improving the efficiency and the quality of metadata integration in DWs. The simple use of CWM by different tool vendors is defined as a passive approach by the authors, meanwhile they consider that an active approach is to design the KBS to use logic rules to reason in two directions, i.e., from the specific metadata models to the generalization metamodel and vice versa. In the proposed architecture, the KBM serves as a central repository for metadata definitions against which any CWM implementing tool is validated.

Independently of the integration technique and methodology, the integration systems also propose different kinds of data architectures that in certain contexts yield better results for their intended purpose. A successful architecture to Database integration and distribution has been the *Federated Database* (FDB) approach [Sheth, 1991, Li and McLeod, 1991, Hsiao et al., 1990, Hsiao, 1992a, Hsiao, 1992b, Kamel and Kamel, 1992, Blanco et al., 1994], where the core idea is to partition data at the logical level, at the physical level, or both. The logical division implies the division of databases into components for allow-

ing separate control over each component, meanwhile the physical partition concerns the allocation of data for storage to the nodes of a network [McLeod and Heimbigner, 1980]. The approach is also called a *decentralized database*. While the FDB approach is a good design solution for solving physical distribution, imposing access policies and others kinds of constraints, it poses some issues when users need to query a federation of databases when the underlying DBs are incompatible [Czejdo et al., 1987]. The best solution, like in data integration, is to define a transparent logical layer implementing query rewriting mechanisms (for example, [Wyss and Van Gucht, 2001]) that offers end users the possibility to seamlessly query the federation of databases.

The FDB approach has been adapted for the DW environment, where the *Federation of DataWarehouses* (FDW) [Kern et al., 2011, Maleszka et al., 2012, Mouhni and Elkalay, 2013, Kern et al., 2013, Cabibbo and Torlone, 2005, Banek et al., 2006, Akinde et al., 2002] has been proposed for logically and physically dividing independent DWs and DMs, or to provide integration architectures for heterogeneous DW environments. The work in [Berger and Schrefl, 2008] proposes such an architecture for *logical* DW integration, as opposed to low-end integration (called *physical* integration in the paper). The FDW architecture distinguishes among four different layers: the *application* or *presentation* layer uses the global schema residing on the *federation layer*; the *intermediate* wrapper layer defines the export and import schemas from the native schemas of the component systems, that in turn reside on the *foundation* layer (see Figure 2.8[3] for a detailed architecture). A query on the global schema is analyzed, decomposed into sub-queries and forwarded to the component systems. Upon receiving the responses, the result data is consolidated and transformed to the global schema in the canonical model again [Berger and Schrefl, 2008].

Apart from the proposed architecture, the authors provide other important contributions to DW integration, like identifying key requirements for the FDW integration architecture and proposing the *dimension repository* as part of the FDW architecture that replicate and consolidate dimensions. On the contrary, *fact* data is maintained in the original repositories, and is integrated only at query time. For the integration purposes, the authors rely on semantic mappings expressed through *Dimension Algebra* (DA) [Cabibbo and Torlone, 2005] expressions that are easily maintainable in an incremental way, and propose the *Fact Algebra* (FA) for *facts* integration.

Although the work in [Berger and Schrefl, 2008] presents a solid framework for DW integration, the implementation is manual, and requires consistent development prior to implementation. On the other hand, the current thesis aims at automating the mapping discovery and integration phase, and is

[3]The figure was extracted from [Berger and Schrefl, 2008]

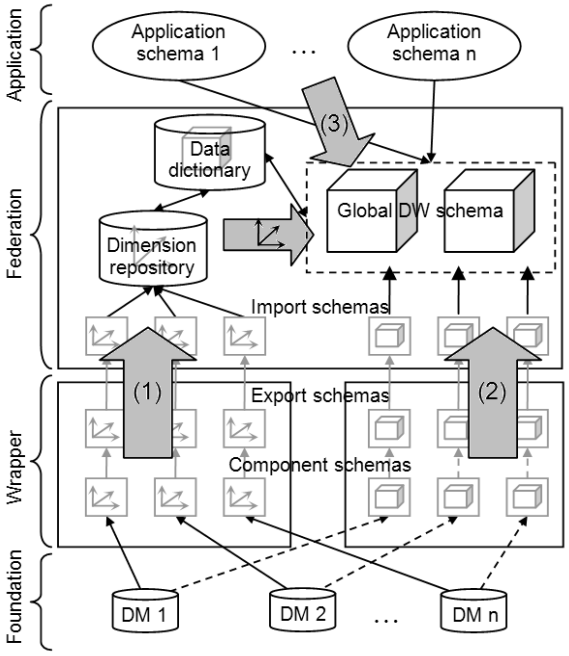


Figure 2.8: The Federated DataWarehouse Architecture

architecture-independent. This means that the mapping discovery approach can be successfully used for a federation of heterogeneous DW as well as other types of DW integration architectures.

In [Berger and Schrefl, 2006], the authors analyze *Multi Datawarehouse Systems* (MDWHS) and divide DW integration architectures among *loosely coupled* and *tightly coupled* architectures. The first type is referred to virtual integration, similar to the GAV approach, and the resulting DW is not materialized. The authors note that this approach is advantageous for ad-hoc queries and when the permanent buildup of a common distributed DW is impossible. On the contrary, a tightly coupled integration means that the final DW is materialized, which in stead allows to easily integrate additional system components, such as OLAP server and OLAP query tools to facilitate the analysts' work [Berger and Schrefl, 2006]. Furthermore, the paper proposes an integration approach for loosely coupled architectures by means of the SQL-MDi (*SQL for multi-dimensional integration*) query language and analyzes DW schema and instance heterogeneities at the fact and dimension level.

As in the previous approach, the main issue is that the method relies heavily on developers that have to manually provided mappings between dimension levels, rename ambiguously named attributes, convert attribute domains in dimension tables, merge overlapping dimension instance sets, resolve naming

conflicts of dimension instances and choose a roll-up hierarchy to use in the virtual data cube [Berger and Schrefl, 2006]. In the approach later presented in this thesis, these requirements are automatically executed or they are not required.

In [Schütz and Schrefl, 2011] the authors propose an integration approach based on the *Hetero-Homogeneous Hierarchies* model for dimension hierarchies and cubes with inherent heterogeneities [Neumayr et al., 2010]. A hetero-homogeneous hierarchy is a hierarchy with a single root node that is (1) homogeneous in regard to a minimal common schema shared by all sub-hierarchies, where a sub-hierarchy is a hierarchy rooted in a child of the root node, (2) heterogeneous in regard to the specialized schemas of sub-hierarchies [Neumayr et al., 2010]. The dimensions are modeled by means of *multi-level objects* (m-objects) and *multi-level relationships* (m-relationships)[4], that the authors previously defined [Neumayr et al., 2009]. Although the *homo-heterogeneous* approach is useful for representing distinct dimensions (thus heterogeneous) for different departmental needs, the dimensions are also confined to be a part of a known, well-defined macro structured, modeled by means of *m-objects*.

For integrating new dimensions into the *hetero-homogeneous* model, the authors propose an object-oriented approach, on which the *m-object* concept is based: an iteration of insertion and reorganization. First, the the highest abstractization of the *m-object* defining the new dimension is identified by overlapping the attributes of the *m-object* with the attributes of already existent *m-objects*, then the new *m-object* is iteratively compared to specialization of the identified *m-object* to find a more specific place in the dimension hierarchy for inserting it. This approach serves for resolving semantic heterogeneity by inserting new dimension into the existing *hetero-homogeneous* model.

For resolving *syntactic* heterogeneity, the authors suggest the use of mapping rules that "assert semantic equivalence between m-objects of different dimensions, or between levels and attributes of different m-objects". However, no mapping rule definition is defined, nor a way to automatically derive them. It is thus safe to assume that in the proposed methodology the mappings have to be manually specified by developers. The main difference between this approach and the one proposed in the current thesis is that the methodology proposed in this thesis automatically computes the semantic mappings and, depending on the integration architecture, the dimensions resulting from the integration step are always homogeneous. As will be shown in the following chapters, this has a great impact on guaranteeing *summarizability* properties to the resulting dimensions, apart from minimizing the instance integration ef-

[4] *m-objects* and *m-relationships* provide a natural, intuitive representation of the *concretization* of objects and relationships along multiple levels of abstraction

fort.

The works in [Cabibbo and Torlone, 2004a, Cabibbo and Torlone, 2004b] make use of a multidimensional conceptual data model (called $\mathscr{MD}$ [Cabibbo and Torlone, 1998]) for reasoning on heterogeneous dimensions. The model is extended with a *dimension algebra* (DA) that is composed of three operators: *selection*, *projection* and *aggregation*. The three operators are used for creating DA expressions that express possible relationships among dimensions, namely *comparability* and *compatibility*. Furthermore, when two dimensions are comparable, the authors define the *intersection* of dimensions through DW expressions. Furthermore, the authors investigate the impact of dimension compatibility on drill across queries.

This setting is interesting for analyzing and proposing dimension integration, as dimension *comparability* (and *compatibility*) can be used to analyze whether two dimensions are similar, and the *intersection* may be used as a starting point for composing a common super-dimension entailing the two initial dimensions. That is the case in [Torlone, 2008], where the author tackles the DW integration problem by starting with dimension integration. As in other approaches, the proposed integration methodology uses dimension mappings, expressed as partial, injective function that identifies similar/identical dimension attributes of two distinct dimensions. Moreover, the author of the paper investigate three quality properties that a matching may have, namely *coherency*, *soundness* and *consistency* that describe how similar are the matched dimensions. Of course, the lesser the quality of the mapping means that it will produce low results when used for integration purposes. The dimension mapping and quality properties will also be used in the current thesis, and the preservation of these properties after the integration step will be analyzed and proved.

Furthermore, in [Torlone, 2008] the author proposes two different approaches to data mart integration, a loosely and a tightly coupled approach. The first case is used for identifying similar elements in two different data marts and for allowing the execution of drill-across queries while preserving the independence of the original sources. No common, integrated dimension is created in this case. The second approach creates a materialized view by combining data from the two data sources. This later case produces identical results to the method proposed in the current thesis, in one specific case: if the bottom attributes of the two dimensions contain identical members and are correctly mapped. One final note for the work in [Torlone, 2008] is that the mappings between heterogeneous dimensions have to be manually specified, as well as the dimension mapping properties. As stated before, the method proposed in the current thesis automatically computes semantic mappings, while the preservation of the quality properties will be proved.

A work similar to the mapping discovery step that will be later described was presented in [Banek et al., 2007, Banek et al., 2008], where the authors propose a method for mapping distinct DW elements by means of a semantic similarity function based on semantic matching approaches previously proposed in literature [Bergamaschi et al., 1999, Rodríguez and Egenhofer, 2003, Dhamankar et al., 2004]. The similarity score is computed first for distinct elements by using the attribute name similarity (*nsim*) and empirically proposed data type compatibility *tcoeff* (the method considers only four basic data types: *numeric*, *string*, *datetime* and *boolean*) that are modulated by means of an empiric coefficient *texp*. As DWs have a complex, multidimensional structure, the authors then calculate a structure similarity coefficient by considering the similarity score of the elements composing the two schemas and by adopting the approach proposed in [Madhavan et al., 2001].

The individual dimension elements are mapped by considering dimensions as graphs and applying the *Stable Marriage* for bipartite graphs problem solution as mapping criteria [Lovász and Plummer, 1986, Melnik et al., 2002]. For the matching purposes, the authors also discard mappings that violate the partial order in the hierarchies (i.e., mappings that are not *coherent*, as defined in [Torlone, 2008])

This later work shares some similarities with the mapping discovery step proposed in the present thesis, but rather than using semantics for generating the mappings, it will be used for pruning the mappings discovered by means of structure and cardinality similarity. Moreover, incoherent mappings will not be discarded, rather the current methodology proposes mappings that are already coherent. Furthermore, the methodology proposed in [Banek et al., 2008] is limited only to mapping discovery, whereas the aim of this document is to provide a complete dimension integration methodology comprising schema mapping, schema and instance integration.

Chapter 3

Heterogeneous Data Warehouse Analysis

3.1 Preliminaries

Many models and approaches have been proposed for the conceptual design of a DW and for deriving multidimensional schemata from E/R [Chen, 1976] or relational schemata (e.g., [Golfarelli et al., 1998a, Golfarelli and Rizzi, 1998, Cabibbo and Torlone, 1998]), XML (e.g., [Golfarelli et al., 2001]), ontologies [Khouri and Ladjel, 2010], web [Zhu and Buchmann, 2002] and other structured/semi-structured data stores (see [Romero and Abelló, 2009] for a survey and [Rizzi et al., 2006] for a critical discussion on DW design and modeling research), although most of them share the same basic ideas of data (*facts*) organized along *dimensions* of analysis that are used to interpret the information at various levels of granularity. The facts are analyzed using numerical *measures* that express a property of interest.

The *multidimensional* view of data fits well the needs of analysts and developers, and it represents an intuitive method for conceptually and graphically representing the concepts of interest both for developers as for business people. The DW dimensions are represented as hierarchies of aggregation levels, usually called *categories* [Hurtado and Mendelzon, 2002] or *dimensional attributes* [Golfarelli et al., 1998b] that are populated by *members* (or values). For example, in a *temporal* dimension, like the one in Figure 3.1, *year* is a category, while {2010, 2011, 2012} are members of the category year.

For the purpose of this thesis, the formalization proposed in [Hurtado et al., 2005, Hurtado and Mendelzon, 2001] will be used to reason both on dimension schemas and instance properties. The dimension mappings will be expressed both in this formalism, as in the *Dimensional Fact Model* (DFM) that allows

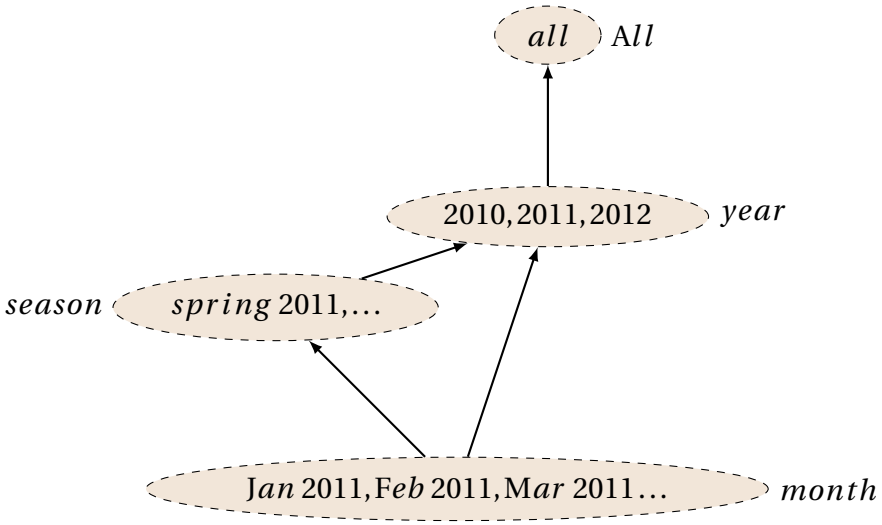


Figure 3.1: A *time* dimension

a more detailed specification of the mapping types between dimensional attributes. The mappings expressed in DFM may also be used for query reformulation, as described in [Golfarelli et al., 2011].

In [Hurtado et al., 2005], a *dimension schema* is a *directed acyclical graph* (dag) H = (C, $\nearrow$), where C is a finite set of *categories*, having a distinguished category $All \in$ C that is a *sink*[1]. The partial order relation $\nearrow$ expresses the conceptual aggregation relations among categories, also called *roll-up* relations.

The dimension schema may also be represented as a directed graph, where the partial order relation $\nearrow$ is a subset P $\subseteq$ C $\times$ C, where $(c_1, c_2) \in$ P if and only if $c_1 \nearrow c_2$. The relation $c_1 \nearrow c_2$ states that c_1 *rolls-up* to c_2, or that c_2 expresses more aggregate information than c_1.

A *bottom* category is a category c_{bottom} that is reachable from no other node of the graph through an arc or a path (i.e., there is no $c \in$ C such that $c \nearrow c_{\text{bottom}}$). For the purpose of this thesis, a single bottom category for dimension will be assumed. Note that various models that allow a dimension to have more than one bottom category (e.g. [Jagadish et al., 1999]) have been proposed in literature; however dimensions with only one bottom category allow a cleaner representation of multidimensional data, although in some cases it implies data replication and redundancy. For simplifying the reasoning about schema and semantic mappings properties, single bottom category schemas may be assumed without falsifying the generality of the statements and the reasoning itself.

[1] A *sink* is a vertex of the graph that is reachable from every other node of the graph through an edge or a path.

An example of dimension schema is presented in Fig. 3.1, where [$month \nearrow season$] and [$month \nearrow year$].

A *hierarchy domain* is a dag $h = (M, <)$, where M is the set of *members* of the hierarchy, with a distinguished member $all \in M$ that is a sink. The members are also organized in a graph structure ($<$ is a partial order relation on the set M) to express the roll-up relations among the categories they belong to. In Figure 3.1 we have, for example, that $Jan\ 2011 < 2011$ and that $Spring\ 2011 < 2011$.

The *hierarchy domain* is the instance data of the *dimension schema*. The relation between the two concepts is expressed through the definition of the *dimension*. A dimension d over a schema $(C, \nearrow)$ is a graph morphism between the schema and the instance, which is a function $d : (M, <) \longrightarrow (C, \nearrow)$ such that:

a) $d(all) = All$

b) $\forall x, y, z$ such that $x <^* y$, $x <^* z$ and $y \neq z$, then $d(y) \neq d(z)$. [2]

Condition *b*) states that a category member may roll-up to two or more members of *distinct* categories. To illustrate the condition, in the example of Figure 3.1 the member $Mar\ 2011$ rolls-up to members $spring\ 2011$ (category $season$) and 2011 (category $year$) that belong to different categories. It cannot be that the member $Mar\ 2011$ rolls-up to two distinct members of the same category (for example, $year$), as this would violate the $1 : n$ relation that characterizes the roll-up relations among dimension members.

Let $\mathfrak{m} : C \rightarrow \mathscr{P}(M)$ be a function that assigns each category the set of members of that category; in other words $\mathfrak{m}(c) = \{m \in M | d(m) = c\}$, for every $c \in C$. We will occasionally call $\mathfrak{m}(c)$ as the "members of c". The hierarchy domain may also be described by a family of *partial* ot *total* roll-up functions [Cabibbo and Torlone, 2004b] $\rho^{c_1 \rightarrow c_2} : \mathfrak{m}(c_1) \rightarrow \mathfrak{m}(c_2)$; the roll-up functions are defined for all $c_1 \nearrow c_2$ as $\rho^{c_1 \rightarrow c_2}(m_1) = m_2$ for all $m_1 \in \mathfrak{m}(c_1)$ and $m_2 \in \mathfrak{m}(c_2)$ such that $m_1 < m_2$.

3.2 Homogeneous and Heterogeneous Dimensions

Studying the DW integration problem, it is interesting to analyze the different kinds of *heterogeneities* that sources may contain and the way they are eliminated/introduced by applying the integration methodology.

Heterogeneity is generally used to define systems that are in some way different, with distinct data/information or different ways of representing the

[2] $<^*$ is the transitive and reflexive closure of $<$

same information. As there is no general consensus as how to classify homogeneity/heterogeneity, we may distinguish two types that apply in DW integration: *intra-* and *inter-schema* heterogeneity.

3.2.1 Intra-schema Heterogeneity

Intra-schema heterogeneity is used to define roll-up inconsistencies inside a single schema. In [Hurtado and Mendelzon, 2001] a schema is defined *homogeneous* if and only if all the members in a category c_1 roll up to one single member of every category c_2, if $c_1 \nearrow c_2$. In other words, if a member m_1 of the category c_1 rolls-up to a member of category c_2, then all members of category c_1 roll-up to a member of category c_2.

Formally:
$\forall c_1, c_2 \in \mathrm{C} : c_1 \nearrow c_2$, if $\exists m_1 \in \mathfrak{m}(c_1), m_2 \in \mathfrak{m}(c_2) : m_1 < m_2$, then
$\forall m_i \in \mathfrak{m}(c_1), \exists m_j \in \mathfrak{m}(c_2) : m_i < m_j$.

Furthermore, if a schema is homogeneous and has only one bottom category (like the ones assumed throughout the current thesis), then it is called *strictly homogeneous*. If a schema is not homogeneous, then it is called *heterogeneous*. We will refer to this kind of homogeneity as *intra-schema homogeneity* (the opposite being *intra-schema heterogeneity*).

Initial multidimensional models were homogeneous, but the restriction has been later dropped to allow a more compact dimension design and a more efficient space allocation due to the lower number of categories [Hurtado and Mendelzon, 2001]. This allows a more efficient physical allocation of data, and the possibility of increasing data management efficiency by avoiding data replication and redundancy and by optimizing data access.

On the other hand, intra-schema homogeneity allows a clear representation of a hierarchy schema as it provides an intuitive representation of the aggregation levels inside one single dimension. In fact, for every two categories c_1 and $c_2 \in \mathrm{C}$, a roll-up relation $c_1 \nearrow c_2$ clearly states the completeness of the aggregation function, as every member of c_1 is related (or aggregated) to a member of c_2. This implies that the roll-up function $\rho^{c_1 \to c_2}$ is a *total* function for all c_1 and c_2. For analysts that handle the schema it is easier to interpret the relations among aggregation levels, as to represent them.

Intra-schema homogeneity is also important when analyzing dependent `GROUP BY` queries. In [Harinarayan et al., 1996], two queries Q_1 and Q_2 are dependent if the results of one can be computed from the results of the other. For example, when analyzing the sales of one company, if every sale is related to a *city* and every city belongs to a *region*, then the total revenue for the sales grouped by *region* (call it query Q_2) may be obtained from the revenue grouped

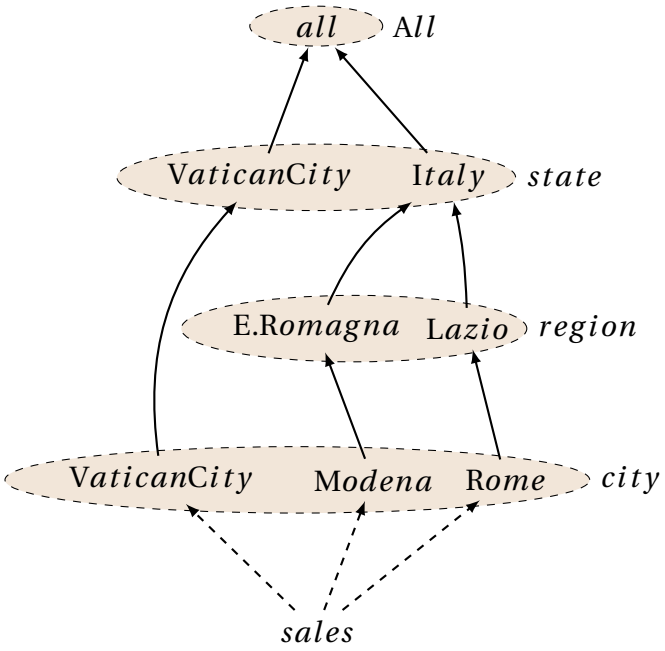


Figure 3.2: Heterogeneous dimension

by *city* (query Q_1) by adding the revenue for every city inside one single region. This observation is important when performing optimization on the DW by materializing frequently accessed views [Harinarayan et al., 1996] or by precomputing aggregations when using the `CUBE` operator [Gray et al., 1997].

The reasoning behind dependent `GROUP BY` queries may be considered as a problem of *completeness* of the aggregation data computed at various aggregation levels. In Figure 3.2, the sales revenue may be computed for each *city* and for each *region*. The revenue grouped by region may be obtained from the revenue grouped by city, however this query will not provide the user the total revenue stored in the DW, rather the one that is related to a specified *region* (the revenue for the cities without a region will not be considered). This is because not every city has a corresponding region (in the example, *Vatican City*).

Although the results of Q_2 may be obtained from Q_1 by aggregation, the result of the query "the total revenue group by *state*" (query Q_3) may be obtained only from query Q_1, and not Q_2, because the schema is *heterogeneous*. Had the schema been *homogeneous*, then Q_3 could also have been compute from Q_2. In a homogeneous dimension, a query Q_{m_j} depends on every query Q_{m_i} if and only if $d(m_i) \nearrow d(m_j)$.[3]

[3]Assuming m_i and m_j are members of some category, and Q_{m_i} is an aggregated query grouping the measures by m_i (and Q_{m_j} by m_j respectively).

Intra-schema homogeneity is also a necessary condition for summarizability which has been defined in the field of *statistical databases* as the ability to correctly compute aggregated data according to a classification node (or C-node) from the same pre-aggregated data at a lower C-node [Rafanelli and Shoshani, 1990].

Summarizability is the ability to correctly compute aggregate data from other aggregate data. It is a concept similar to the dependent `GROUP BY`, and it adds the *completeness* assumption. To intuitively explain summarizability, consider the example in Figure 3.2. The results of query Q_3 cannot be obtained from query Q_2, but only from query Q_1. This is because if the total revenue is grouped by city (Q_1), and then by region (Q_2), then query Q_2 will yield only the revenue of the cities that have a region (excluding $VaticanCity$, because the schema is heterogeneous). Furthermore, by computing the results of query Q_3 from query Q_2, the result of the query would be different than by computing it from query Q_1, or by computing it directly from the individual data.

Another formalization for the dependency of summarizability on homogeneity has been provided in [Lenz and Shoshani, 1997], where the authors define three necessary conditions for guaranteeing summarizability: *disjointedness*, *completeness* and type compatibility. The first two conditions are equivalent to homogeneity, thus homogeneity is a necessary (not *sufficient*) condition for allowing summarizability of multidimensional data [Beneventano et al., 2013].

Furthermore, by using the *closed-world* assumption (part of the *completeness* hypothesis), a homogeneous dimension with only one bottom category is in *dimensional normal form (DNF)*, as defined in [Lehner et al., 1998]. Checking homogeneity can thus assert important quality properties for the final schema and instance after applying the integration methodology.

In general, homogeneity and heterogeneity are not preserved when integrating two distinct DWs. This is also the case when using the integration methodology that will be described in this paper. Examples will prove that it is possible to obtain a heterogeneous dimension from two homogeneous dimensions, or a homogeneous dimension from two heterogeneous dimensions (and the other way around). A sufficient condition for preserving homogeneity will be presented in this thesis.

3.2.2 Inter-schema Heterogeneity

Inter-schema heterogeneity is a concept derived from database theory and used to describe two or more different data sources that are in some way distinct by schema or by form (see [Sheth and Larson, 1990, Lenzerini, 2002] for a discussion about heterogeneities).

Following the same approach, two or more different DWs may contain the same information, but structured differently or represented by different values. The heterogeneities may be at instance-level (e.g., a *month* may be represented by its name or by a number) or at schema-level (e.g., hierarchies may contain similar information but structured differently, like higher granularity, inconsistent roll-up functions, different category names; the same information may be represented as a fact attribute in one schema or as a category in another; or the schemas may contain different and possibly inconsistent measures that are incompatible/inapplicable on all the DWs).

An intuitive classification of the schema and instance conflicts is presented in [Torlone, 2009], where the conflicts are divided by dimension and fact conflicts, as well as by schema and instance conflicts:

- *Dimension Conflicts*
  - Schema: conflicts can arise on entity names (e.g., different names for the same dimensions and/or different names for similar levels of two dimensions) and on dimension hierarchies (similar dimensions organized over different levels of aggregation and/or inconsistencies on the roll-up relationships between levels).
  - Instance: conflicts can arise on member names (different names for the same members of different dimensions) and on the members of dimensions (similar dimensions populated by different members).
- *Fact conflicts*
  - Schema: conflicts can be encountered on names (different names for the same measures) and on dimensions that differ in number and/or in the levels of aggregation.
  - Instance: conflicts can arise on measures (inconsistent values for the same measures and/or differences in scales).

Other researchers (for example, [Berger and Schrefl, 2006, Banek et al., 2008]) have analyzed and provided more detailed classifications of the different kinds of heterogeneities that may occur between two distinct DWs (for example, Table 3.1 [Berger and Schrefl, 2006] provides such a classification).

Facts conflicts will not be taken into discussion in this thesis, as it considers only dimension integration and the conflicts it may eliminate during the integration steps.

Without analyzing *inter-schema heterogeneity* any deeper, it is important to note that although there is a general, intuitive notion of heterogeneity, until

Table 3.1 Inter-schema heterogeneities

	Facts	Dimensions
Instance level	– Overlapping facts – Disjoint fact subsets	– Naming conflicts – Overlapping members – Heterogeneous roll-up mappings
Schema Level	– Different number of dimensions ("dimensionality") – Naming conflict (measures) – Domain conflict (measures)	– Diverse aggregation hierarchies – Inner level domain conflicts – Lowest level domain conflicts – Domain and/or naming conflicts (non-dimensional attributes)
Schema vs. Instance	Fact context as dimension instances	Dimension members as contextualized facts

now there has been no common acceptation of standard dimension and facts conflicts, nor a widely accepted solution to eliminating these kinds of heterogeneities. Most solutions provided in literature so far point out individual inconsistencies, and in some cases provide *ad-hoc* solutions to eliminate them.

Inter-schema heterogeneity should also be eliminated by means of semantic mappings, that provide a solid solution not only for expressing semantic heterogeneity, but also for potentially eliminating it (in cases where this is actually possible).

The focus of the mappings should not be on the data actually contained in the source repository, rather on the type of information that is modeled by the schema of the data repository. To make this distinction clear, suppose the two *time* dimensions from Figure 3.3 (extracted from Figures 1.1&1.2) contain data from two distinct time periods, say 2006-2010 and 2011-2013. Even though the intersection of the two instances produces a void set, the two dimensions are still similar, as they contain similar dimension categories.

Among the main intra-schema heterogeneities, one can note at schema

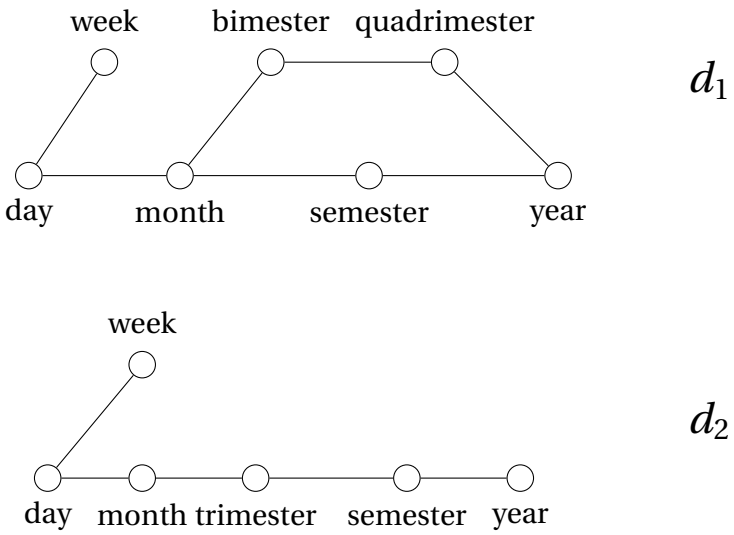


Figure 3.3: Time dimensions

level the following conflicts (as classified in Table 3.1):

- *dimension attributes naming conflicts*: all categories attributes have different labels, although they refer to the same concepts (e.g., *dy* = *day*, *mth* = *month*); unfortunately, it is very common in Data Warehousing that developers use different labels, abbreviations, or worse still, synthetic labels with no linguistic meaning (e.g., *attr2*, *attr2*, *attr3*, etc.) to denote common attributes. An automatic matching approach should be able to correctly detect and match similar dimension categories, regardless of their label. That is the reason why in the current approach semantics (that relies on the categories' labels) is used only for validation purposes, not for matching discovery;

- *inner level domain conflicts*: the *roll-up* hierarchies present different aggregation levels, that are not necessarily inconsistent. In dimension d_1, for example, a month is aggregated by *semester* and then by *year*, while in dimension d_2 a month is aggregated by trimester, then by semester and finally by year. Dimension d_1 also presents a different aggregation path between month and year (*month-bimester-quadrimester-year*). These aggregation paths, although different, are not conflicting. They simply express the different requirements of different working groups for representing the same data. An automatic matching methodology should, however, be able to correctly identify these differences and propose mappings that are correct, meaning that they are able to both identify the rela-

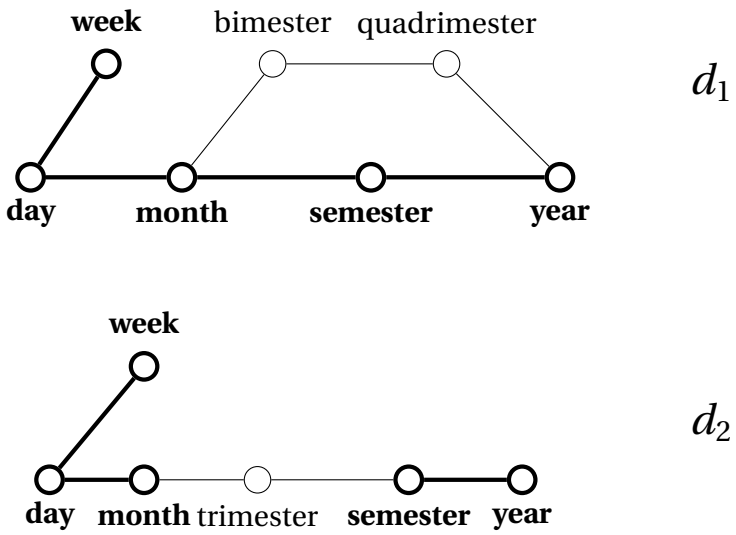


Figure 3.4: Time dimensions - common categories

tions between identical dimension categories as to express different (yet coherent) relations between distinct dimension categories.

Summing up, in order to resolve the two above described inter-schema heterogeneities, a correct mapping should, first of all, identify that:

- $d_1.day \leftrightarrow d_2.day$
- $d_1.week \leftrightarrow d_2.week$
- $d_1.month \leftrightarrow d_2.month$
- $d_1.year \leftrightarrow d_2.year$

Figure 3.4 highlights the common aggregation levels of the two distinct categories.

In the mapping and integration approach described in the current thesis, inter-schema heterogeneity is implicitly reduced or eliminated by performing dimension integration. Eliminating all schema and instance conflicts is outside the scope of this thesis, as such a goal is so vast and context dependent that is almost impossible to propose a universal solution for all DW integration architectures.

One interesting case discussed in this thesis (Section 5.2) allows the complete elimination of inter-schema heterogeneity, by rendering two dimensions identical after performing the integration methodology. In particular, naming

conflicts and overlapping members will be handled due to the use of the *RELEVANT* clustering technique, while the presence of diverse aggregation hierarchies and inner level domain conflicts will be also eliminated in the schema integration step.

Chapter 4

Dimension integration

The current chapter describes a complete dimension integration methodology that is able to identify common dimension categories between distinct, yet similar dimensions, integrate heterogeneous schemas and consistently populate the integrated schemas with correct instance values.

Any general dimension integration approach should follow these general steps [Torlone, 2009][1]:

1. **identification** of the dimensions of the facts that can be combined to perform the integration,
2. **resolution** of conflicts between common dimensions,
3. **reconciliation and integration** of dimensions to the desired level of interoperability.

This is also the case of the methodology that will be discussed in the current chapter, that is divided as in the following steps:

1. *Generate semantic mappings*: an automatic mapping discovery method that uses structural properties for mapping discovery purposes, and semantics for mappings pruning;
2. *Integrate dimension schemas*: the discovered semantic mappings will be used for integrating the dimension schemas. Two possible approaches will be discussed in this section: integration under a distributed network of repositories, and the creation of a central, integrated DW;
3. Integrate instance data: the integrated schema(s) will be populated with consistent data, by using the *RELEVANT* clustering approach. This allows

[1] The steps were reduced only to dimension integration.

to eliminate some of the instance level inconsistencies, like eliminating *null* values and incorrect aggregation paths.

4.1 Mapping Discovery

It is clear from the above description that the semantic mappings play a fundamental role in the methodology. As discussed in Chapter 3, a semantic mapping describes conceptual similarities between related concepts. Depending on the integration architecture, the mappings will be used both for query rewriting as for data integration inside the integration architecture.

In order to discover semantic mappings, one may use classical data-integration approaches, like semantics or structural matching. It has been, however, proven that combined approaches may yield better results than single approaches taken individually [Bergamaschi et al., 2007a]. For this purpose, the mapping creation step of the current approach is divided in two parts:

- *mappings generation*: mappings between dimension categories will be generated by using structural properties, and a property called *cardinality-ratio*;

- *mappings pruning*: the generated mappings will be pruned by using semantics; this acts as a further validation step for the discovered mappings.

4.1.1 Discussion

The mapping generations step is built on the idea that common information is generally structured in the same way even by different working groups, due to the way that general, well understood concepts are perceived and universally accepted by analysts. The most simple example is *time*. In order to prove the efficiency of the mapping discovery method that will be presented in the following, consider that the *time* dimensions schemas in Figure 3.3 contain different data; for example dimension d_1 contains all the days of the years $2006-2010$, meanwhile the dimension d_2 contains all the days from $2011-2013$. This assumption is relevant for proving that a general matching technique should work even in cases where the instance data is disjoint.

Although the instance data is disjoint, some common properties of the time dimension structure are maintained. Supposing that the instance of the dimension d_1 is complete, then it contains exactly 1.825 *days*, 261 *months*, 60 *months*, 30 *bimesters*, 15 *quadrimesters*, 10 *semesters* and 5 *years*. Considering that the *roll-up* functions are $n:1$ relations, then on average a *month* is an aggregation

of more or less 30 *days*, a *week* is on average an aggregation of 7 *days*, and so on.

The second time dimension (d_2) contains all the days of three years (2011, 2012 and 2013). This means that the instance contains 1095 *days*, 36 *months*, 157 *weeks*, and so on. Considering the $roll-up$ functions, a simple computation leads to the conclusion that on average a *month* is an aggregation of more or less 30 *days*, a *week* is on average an aggregation of 7 *days*, and so on. These aggregation averages are common between the two dimensions, although the instance data is disjoint.

This example is intentionally trivial, and is used to highlight how common, well understood concepts are almost always represented in similar ways by different working groups, in order to reflect the common understanding of the concept of interest. The same, however, may be said for other common dimensions, like geographical dimensions (cities organized into districts, grouped into regions, etc.), common organizational division inside the same or similar companies(e.g., companies divided into areas, divisions, working groups, and so on), etc.

The concept may also be observed when representing other kind of information. Consider, as an example, two companies managing health data. The various health conditions will certainly be categorized in a similar manner, in accordance with the common representation of the knowledge of interest (for example, the World Health Organization provides an International Classification of Diseases - ICD[2]).

Other examples may be common product classifications provided by different working groups, like the *United Nations Standard Products and Services Code* (UNSPSC[3]) or the *Electronic Commerce Code Management Association* (ECCMA[4]). The UNSPSC standard, for example, provides a hierarchical classification of products divided by *segment*, *family*, *class*, *commodity* and *business family* that synthesizes a common view of products that fits the most general needs of different organizations. Companies adhering to such standards will, of course, obtain product classification dimensions that are compatible for integration purposes. This example, however, serves also another purpose, that is to highlight that the classification process of concepts is very often common to different groups, whether or not these groups actually use classification standards. One interesting demonstration will be provided in Section 4.1.4 (experimental evaluation), where two product classification dimensions from two completely distinct DWs will prove very similar, although belonging to two

[2] `http://www.who.int/classifications/icd/en/`
[3] `http://www.unspsc.org/`
[4] `http://www.eccma.org/`

completely distinct companies that didn't use any product classification standard. In this case, the proposed mapping generation step proved efficient by exploiting the similar product classification that was inherently applied by the two working groups.

4.1.2 Mappings Generation

The mapping generating step exploits common classification of similar concepts, as discussed in the previous section. For this purpose, consider the example schemas in Figure 3.3, and the instances described in Section 4.1.1. The schemas of the dimensions are $H_1 = (C_1, \nearrow_1)$ and $H_1 = (C_2, \nearrow_2)$, where

$C_1 = \{day, week, holiday, month, bimester, quadrimester, semester, year\}$ and

$C_1 = \{day, week, month, trimester, semester, year\}$

and the directed edges expressed by the relations $\nearrow_1$ and $\nearrow_2$ are as depicted in the figure.

The mapping generation is divided in the following steps:

- represent dimensions as directed labeled graphs;
- use connectivity matrices to compute a maximum rank subgraph that represents a sub-dimension common to both initial dimensions;
- use the common sub-graph to detect similar nodes of the graphs (categories of the dimensions);
- use inference rules to generate complex semantic mappings between every categories of the initial dimensions.

The directed labeled graphs in Figure 4.1 represent the dimensions in Figure 3.3. The labels of the edges is the result of the *cardinality-ratio* between the total number of distinct members of two related categories. Formally, given a dimension $d : (M, <) \rightarrow (C, \nearrow)$, for every two categories c_i and $c_j \in C$, the cardinality-ratio among the two categories is defined as[5]:

$$\tau_{c_i \rightarrow c_j} = \begin{cases} \frac{\#\mathfrak{m}(c_j)}{\#\mathfrak{m}(c_i)} & \text{if } c_i \nearrow^* c_j \text{ and } c_i \neq c_j \\ 0 & elsewhere. \end{cases}$$

For the mapping purposes, the transitive closure of the graphs will be used[6]. The reason is that given the chain of roll-up relations among dimension categories, the transitive closure can and must be applied to the cardinality-ratio

[5]The function $\mathfrak{m}$ is as defined in Section 3.1

[6]The transitive closure of a graph is another graph that contains an edge between every two nodes that are connected by a path

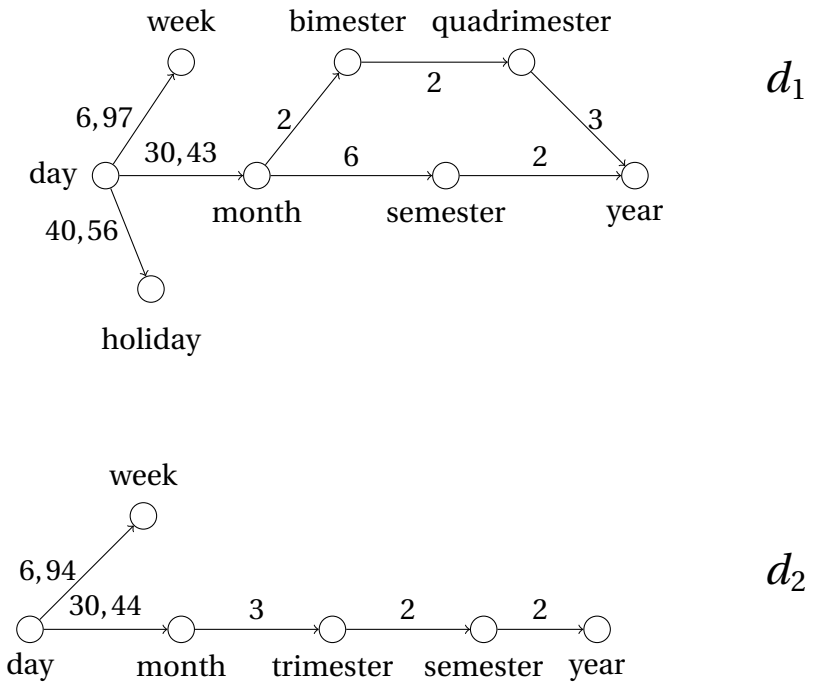


Figure 4.1: Time dimensions

relations among categories. In the dimension d_1, for example, a year is an aggregation of 2 semesters, each being an aggregation of 6 months; by extension, in the instance of the considered dimension, a year is an aggregation of 12 months[7]. The same cardinality-ratio can be noted in the second dimension (d_2), where a year is an aggregation of 2 semesters, a semester is an aggregation of 2 trimesters and a trimester is an aggregation of 3 months. In the instance of the two dimensions, the observation that a year is an aggregation of 12 months may be deduced only by considering the transitive closure of the roll-up relation.

The two dimensions are assigned a connectivity matrix, where the elements composing the matrix are defined as follows:

$$a_{c_i \to c_j} = \begin{cases} \tau_{c_i \to c_j} & \text{if } c_i \nearrow^* c_j \text{ and } c_i \neq c_j \\ 1 & \text{if } c_i = c_j \\ 0 & elsewhere. \end{cases}$$

In order for the method to be effective, the lines and columns of the matrix must not violate the $\nearrow$ relation, meaning that if $c_i \nearrow c_j$, then $i < j$ (assuming that c_i was assigned the i-th row, and c_j the j-th row of the matrix).

[7] In an incomplete instance, this ratio may vary.

Algorithm 1 Approximative common subgraph computation

1: C = { empty matrix }
2: define an error ε
3: **for** every square sub-matrix S_{D_1} of D_1 **do**
4: **for** every square sub-matrix S_{D_2} of D_2 **do**
5: **if** $\forall\ i, j$: $d_{1_{ij}}, d_{2_{ij}} \in \left[\left(1-\varepsilon\right)^{\frac{\left|d_{1_{ij}}-d_{2_{ij}}\right|}{2}}, \left(1+\varepsilon\right)^{\frac{\left|d_{1_{ij}}-d_{1_{ij}}\right|}{2}}\right]$ **then**
6: **if** $rank(S_{D_1}) > rank(C)$ **then**
7: C := S_{D_1}
8: **end if**
9: **end if**
10: **end for**
11: **end for**
12: return C

Algorithm 1 is used to compute a subgraph that is common to both initial graphs. The algorithm repetitively checks if square submatrices of the two connectivity matrices are almost identical. The comparison is being done for each element $d_{1_{ij}}$ and $d_{2_{ij}}$ of the connectiviy matrices D_1 and D_2 (shown in Figure 4.2), and is using a variation ε of the average of the two considered elements, rather than direct equality. This is because in real DWs it is difficult to encounter perfectly identical instance data, due to the heterogeneity of the instances itself. For example, if the instances of the two time dimensions contained complete years, then every year would be composed of twelve months. If, however, the first instance contained four yealf and a half (rather than five), then the cardinality ratio between the members of the category would be different than in the instance of the second dimension. Admitting some variability in the instance (modulated by the variable ε) thus takes into account this heterogeneity and increases the chances of correctly identifying similar categories.

The value of the error modulation variable ε should be manually provided by analysts. A higher value of ε increases the chances of computing a mapping among dimensions, however decreases the probability of finding correct mappings. Obviously, there is no standard threshold value for ε and it must be chosen with care. However, a mapping with a low value of ε involving many of the categories of the dimensions (i.e., a high percentage of categories of one dimension have been mapped) is a positive sign of a correct mapping.

The algorithm repetitively considers submatrices of increasing size, in order to identify the maximum rank common subgraph of the two initial graphs. This way, larger common category sets will be identified. Of course, a mapping linking more categories between two dimensions has more probability of being

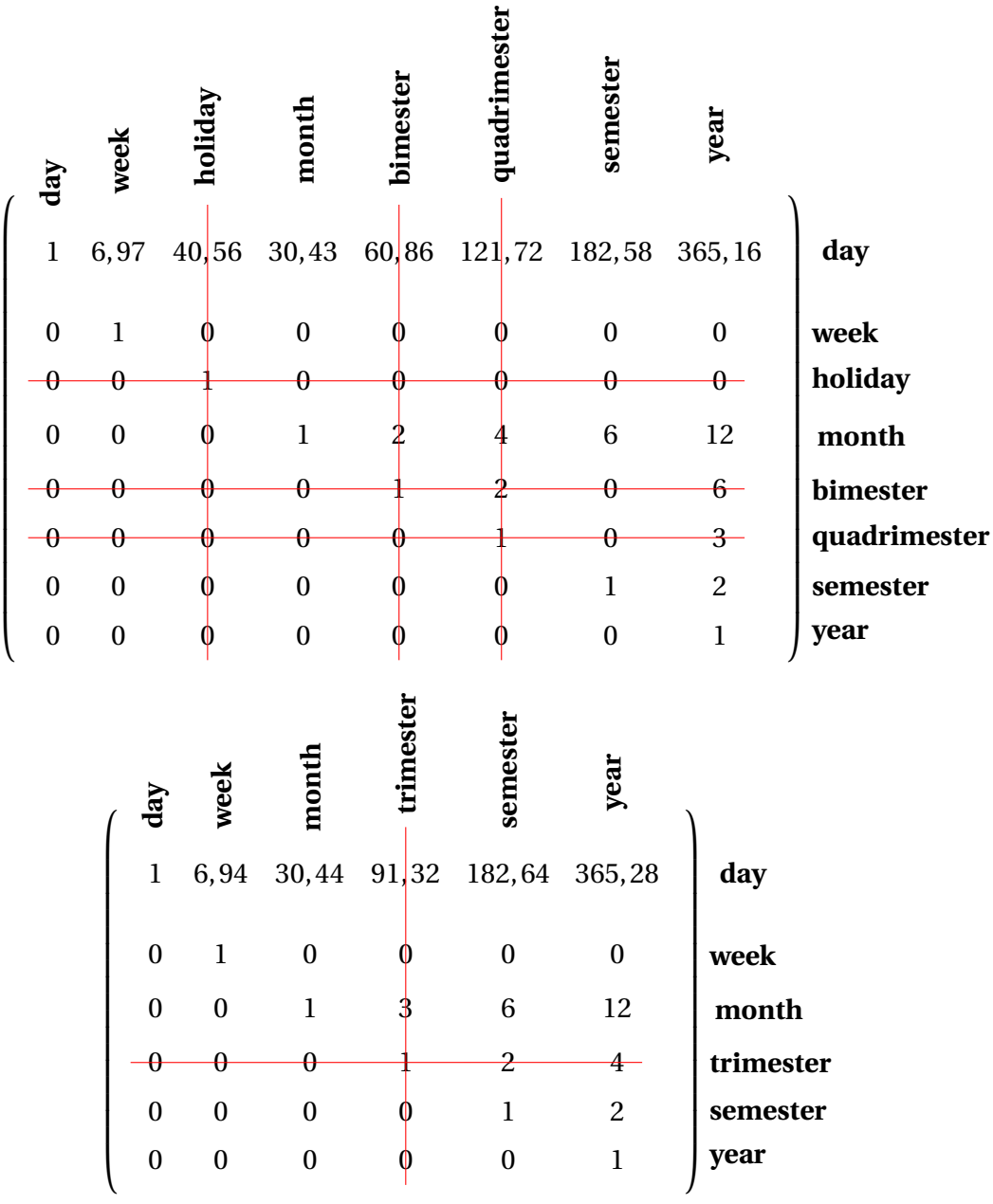


Figure 4.2: Initial connectivity matrices

$$\begin{pmatrix} 1 & 6,94 & 30,44 & 182,64 & 365,28 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 6 & 12 \\ 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

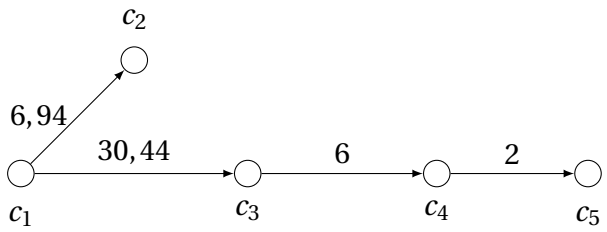


Figure 4.3: Connectivity matrices & Common subgraph

accurate than a mapping connecting only few categories. The approach, however, does not consider the case where more than one maximum rank subgraph exists. Considering though that matrices of the same rank (the maximum one) link the same number of categories of the two initial dimensions (n pairs of categories, where n is the rank of the submatrix), then choosing one maximum rank submatrix over another is not relevant for the mapping discovery.

Moreover, in order for the mapping to be reliable, it is better that the rank of the submatrix is close to the rank of one of the two initial matrices[8].

The result of the algorithm is a connectivity matrix describing a complete labeled graph that represents the parts of the dimension that are common to the two initial dimensions (a concept similar to dimension intersection, as defined in [Cabibbo and Torlone, 2004b]). The common sumbatrix is presented in Figure 4.3, and is obtained from the two initial matrices that are depicted in Figure 4.2. Figure 4.3 also presents a graph that is obtained from the graph described by the resulting connectivity matrix. Note that the matrix describes a complete graph (i.e., the transitive closure of the graph depicted in Figure 4.3). As direct links provide no advantage over already available paths in the dimen-

[8]By computation, it must be that $rank(\mathrm{D}_c) < min(rank(\mathrm{D}_1), rank(\mathrm{D}_2))$, where D_c is the common submatrix

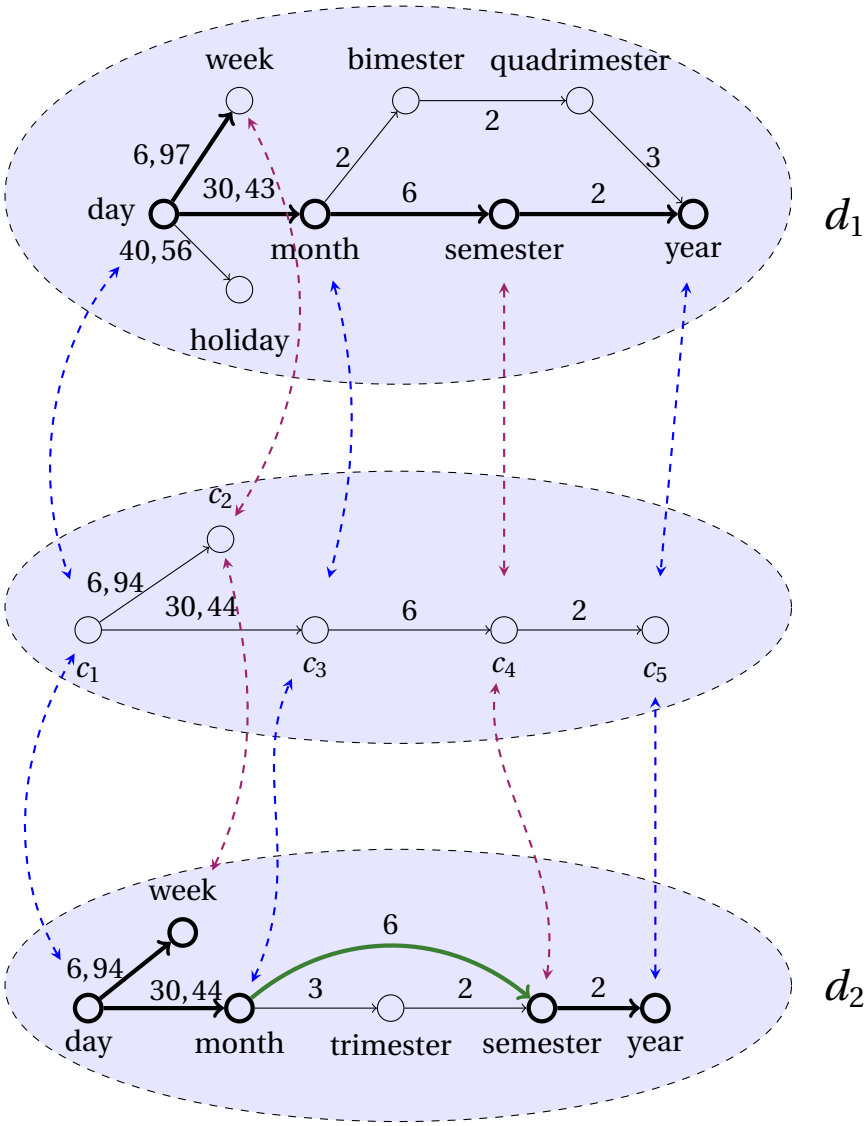


Figure 4.4: Mapping generation

sion hierarchy, they are firstly discarded from the graph. That is, an edge $c_i \rightarrow c_k$ is removed whenever $\exists\, c_j$ such that $c_i \rightarrow c_j$ and $c_j \rightarrow c_k$.

Subsequently, the common subgraph is used to identify pairs of common nodes from the two initial graphs. For example, the first line of the common matrix is obtained from the first line of the first initial matrix and the first line of the second initial matrix (see Figure 4.2). This implies that the categories related to these lines of the matrix are related, so they are mapped. In this case, $d_1.day \Leftrightarrow d_2.day$. The other lines (or columns) of the matrix are used to map

other pairs of the dimension hierarchies. Figure 4.4 shows a graphical representation of the mapped nodes (that are equivalent to the categories of the dimensions). Note in the figure that the green arrow highlights a cardinality-ratio derived from the transitive closure of the direct labeled edges, that was taken into consideration (and maintained) during the mapping process.

In order to formalize the mapping generation step, consider that the dimensions are as formalized in Section 4.1.2:

$d_1 : (M_1, <_1) \rightarrow (C_1, \nearrow_1)$

$d_2 : (M_2, <_2) \rightarrow (C_2, \nearrow_2)$,

where $(M_1, <_1)$ and $(C_1, \nearrow_1)$ are the hierarchy domain and hierarchy schema of dimension d_1 (likewise, $(M_1, <_1)$ and $(C_1, \nearrow_1)$ for d_2). A mapping among dimension d_1 and d_2 is a *partial* function

$\xi : C_1 \rightarrow C_2$

that maps some categories of d_1 to categories of d_2. In the above example, the mapping function is defined as follows:

$\xi(d_1.day) = d_2.day$
$\xi(d_1.week) = d_2.week$
$\xi(d_1.month) = d_2.month$
$\xi(d_1.semester) = d_2.semester$
$\xi(d_1.year) = d_2.year$

4.1.3 Complex Mappings

The mappings generated in the previous section, after being validated as will be shown in Section 4.1.5, may be used for query rewriting and for information integration. Prior to pruning the semantic mappings , it is interesting to analyze a more complex set of mappings proposed in [Golfarelli et al., 2010, Golfarelli et al., 2011]. The authors propose a complex set of five mapping predicates used for OLAP query reformulation in Peer-to-Peer Data Warehousing; only four mapping predicates related to dimension categories, namely *equi-level*, *roll-up*, *drill-down* and *related*, will be considered.

The first mapping predicate asserts that the two *equi-level* attributes have the same semantic and granularity; the *roll-up* predicate states that two attributes have the same meaning, but at a different aggregation level. Also, among two such attributes, there is a $1 : n$ relation. The *drill-down* predicate is the opposite of the *roll-up* predicate, meanwhile the *related* mapping predicate is used to state that the attributes are in a *many-to-many* relationship.

While the first three predicates are straightforward, to explain the last mapping predicate consider that $\{week\}$ *related* $\{month\}$, as a month may have more than one week (it actually has 4-6, considering that weeks start on Monday and

end on Sunday, and that months can contain incomplete weeks) and a week may span across one or two distinct months.

The *equi-level* mapping is identical to the direct mapping expressed through the mapping function ξ previously described. The complete list of complex mappings (as defined in [Golfarelli et al., 2010]) may be obtained by using the following inference rules:

1. **Rule 1**: $\forall\ c_i \in C_1$ and $c_m \in C_2$, if $\xi(c_i) = c_m$, then $\{c_i\}$ *equi-level* $\{c_m\}$

2. **Rule 2**: $\forall\ c_i \in C_1$ and $c_m, c_n \in C_2$ such that $c_m \nearrow_2 c_n$, if $\xi(c_i) = c_m$ then $\{c_i\}$ *drill-down* $\{c_n\}$

3. **Rule 3**: $\forall c_i, c_j \in C_1$ and $c_m \in C_2$, if $c_i \nearrow_1 c_j$ and $\xi(c_j) = c_m$, then $\{c_i\}$ *drill-down* $\{c_m\}$

4. **Rule 4**: $\forall c_i, c_j \in C_1$ and $c_m \in C_2$, if $c_i \nearrow_1 c_j$ and $\xi(c_i) = c_m$, then $\{c_i\}$ *roll-up* $\{c_m\}$

5. **Rule 5**: $\forall\ c_i \in C_1$ and $c_m, c_n \in C_2$ such that $c_m \nearrow_2 c_n$, if $\xi(c_i) = c_n$ then $\{c_i\}$ *roll-up* $\{c_n\}$

6. **Rule 6**: $\forall c_i, c_j \in C_1$ and $c_m, c_n \in C_2$ such that $c_i \nearrow_1 c_j$ and $c_m \nearrow_2 c_n$, if $\xi(c_j) = c_m$, then $\{c_n\}$ *roll-up* $\{c_i\}$

7. **Rule 7**: $\forall c_i, c_j \in C_1$ and $c_m, c_n \in C_2$ such that $c_i \nearrow_1 c_j$ and $c_m \nearrow_2 c_n$, if $\xi(c_i) = c_n$, then $\{c_j\}$ *drill-down* $\{c_m\}$

8. **Rule 8**: $\forall c_i \in C_1$ and $c_m \in C_2$ that have not been previously mapped, then $\{c_i\}$ *related* $\{c_m\}$

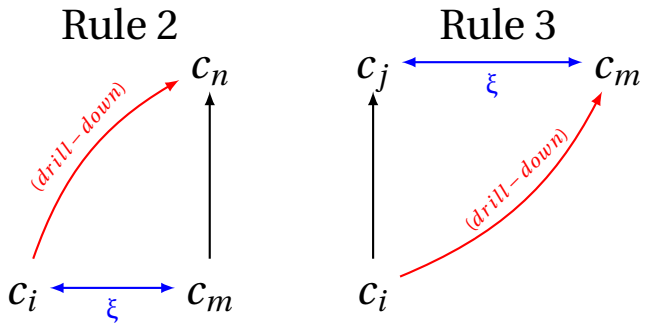


Figure 4.5: Mapping Rules 2 & 3

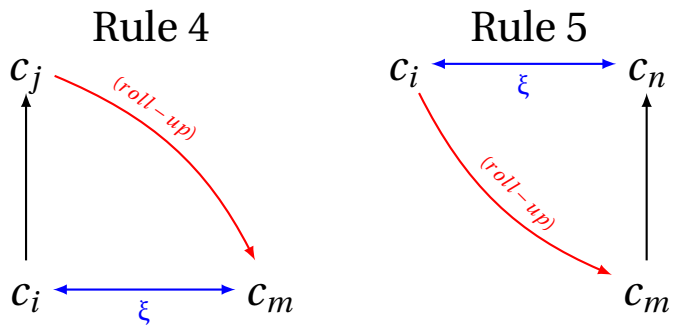


Figure 4.6: Mapping Rules 4 & 5

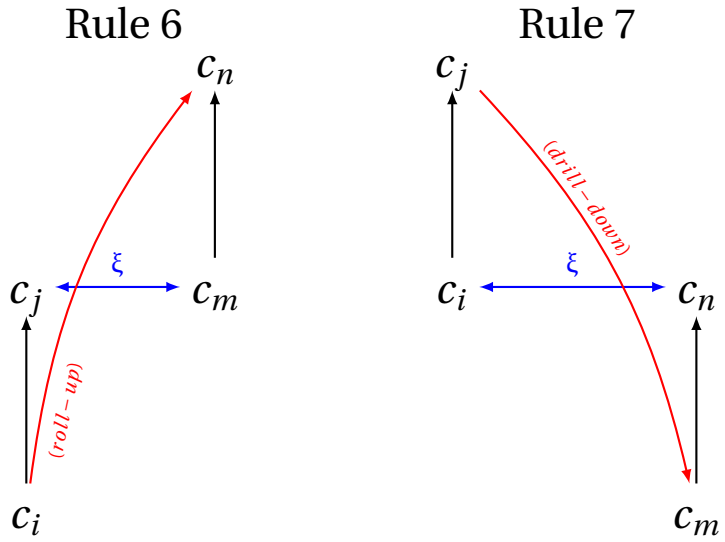


Figure 4.7: Mapping Rules 6 & 7

The rules (that are graphically represented in Figures 4.5, 4.6 & 4.7) are based on the observation that the initial mapping expresses an equivalent aggregation granularity, and they use the roll-up hierarchies present in the initial dimensions ($\nearrow_1$ and $\nearrow_2$) to assert one of the other types of mappings between the individual pairs of unmapped categories. If no mapping has been generated, it is safe to assume that two unmapped categories, although similar, are of no use for integration purposes. That is why they are annotated with a *related* mapping, that will be subsequently validated by using semantics.

4.1.4 Experimental Evaluation

In order to validate the mapping discovery approach, the matching technique was applied to a scenario of three independent DWs. In particular, the sales of three different companies (an *automation technology* company, a *food* producer and a *fashion* designer) were analyzed. The reason for choosing such different companies is to further prove the underlying idea of the approach, which is that people tend to categorize similar concepts in roughly the same way. In this case, the companies sell different types of products or services; however, the mapping methodology is still able to discover identical categories in the dimensions of the different DWs.

The three DWs contain almost 250.000 purchase orders combined, all related to approximately 9.000 customers. The matching methodology was applied to common dimensions in order to verify its efficiency.

Note that for simplicity reasons, the dimension graphs were manually extracted from the DWs, and annotated with the correct cardinality ratio. The error ε was manually computed for each tested case. To automate the computation, a mapping between the multidimensional model and its physical representation must be used (for example, the Mondrian xml schema model[9]). The implementation of the algorithm is trivial under any modern programming language.

The simplest test involved time dimensions that have an almost identical structure, a part from some extra dimensional categories. The expectation was

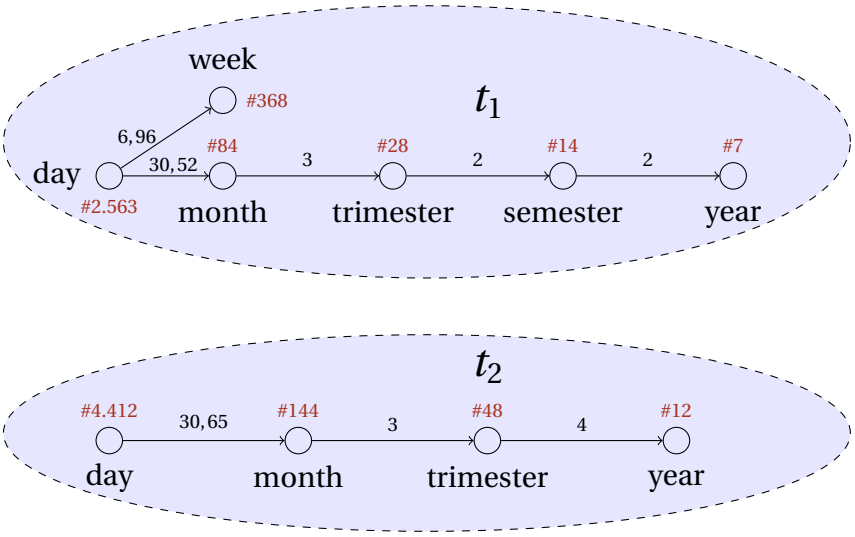


Figure 4.8: Time dimensions of two test DWs

[9] http://mondrian.pentaho.com/documentation/schema.php

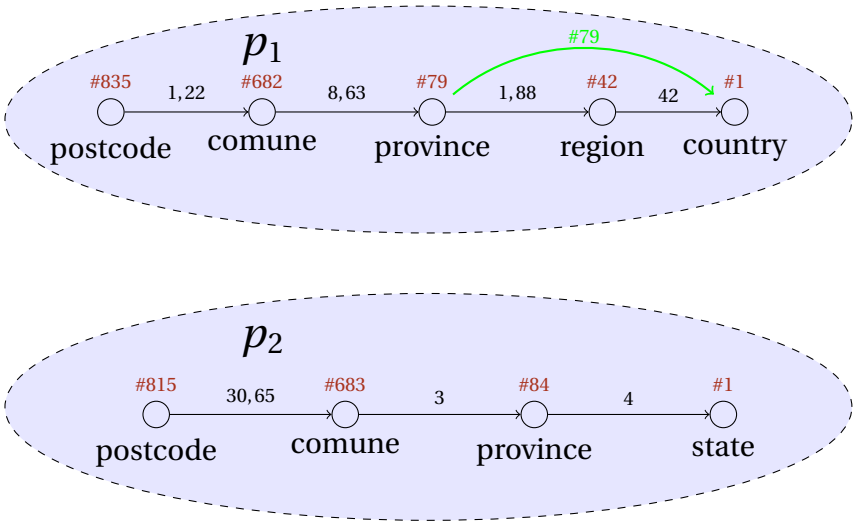


Figure 4.9: *Postcode* dimensions analysis

that the common dimensions matched with a low error ε because time hierarchies have a fixed structure over all DWs. Figure 4.8 contains the description of two of the three time dimensions, with the total number of distinct members of each category (red numbers) as well as the cardinality ratios between various aggregation levels (numbers above the edges). The instance data of the schemas (the sets of members of the base categories) was 75% overlapping, and the common submatrix used to compute the common subhierarchy contained an error $\xi = 4\%$. The methodology correctly identified similar categories of the two distinct dimensions hierarchies.

The second test involved a geographical hierarchy. Unlike the time hierarchy, the probability of having the same structure is lower; however some similarities are maintained. A common subhierarchy was computed from the two original dimensions with an error of $\varepsilon = 7,4\%$. Furthermore, the intersection of the members set was analyzed, and was discovered that the instances contained only 16% of the same data (the other values were totally distinct). This also serves as a proof that a *value*-matching approach would consider the two dimensions with a low similarity, although the dimensions are conceptually identical.

The final test regarded the article dimension hierarchy. As expected, the sets of members of the dimension categories are completely disjoint because the companies sell different product/services. However, the mapping discovery methodology is still able to compute a common subgraph with an error of ε, and subsequently to correctly identify and map related categories.

4.1.5 Semantic Mappings Pruning

Some of the matching approaches for DW elements proposed in literature so far are based on semantics (e.g., [Banek et al., 2008]), which has successfully been used for schema matching (see [Rahm and Bernstein, 2001] for a survey) and data integration (see [Doan and Halevy, 2005] for a survey). Benefiting from the decades of research, semantics may also be used for schema matching in DW environments, although not in the same way. The reasons for not using semantics for mapping discovery are at least two.

First of all, semantics is based on a meaning that is assigned to a label or name of an element or groups of elements, and the relationship of meanings between these members. Although it may provide a solution for DW environments by also considering the relations among elements, the study in this thesis has lead to the conclusion that context specific solutions may provide better results. In fact, as showed in Section 4.4, one context specific solution is to use the well known relation among elements inside every DW, that are grouped into facts that are analyzed according to dimension of analysis. The dimension of analysis are aggregation hierarchies (among the levels of aggregation there is a $1:n$ relationship) structured as directed, acyclic graphs. This *a priori* knowledge may be exploited for schema matching purposes and for yielding better, more accurate mappings among DW elements.

The second reason is that in many cases the names and labels of the elements are meaningless, or are abbreviations with no particular meaning and lack any kind of textual description that is necessary for semantic annotation. In such cases, semantics would prove inefficient for the matching process.

For the above mentioned reasons, in the methodology presented in the current thesis, semantics will be used only for pruning the already discovered mappings. For the validation process, an application of the *Combined WordSense Disambiguation* (CWSD) [Bergamaschi et al., 2007c, Bergamaschi et al., 2007d, Bergamaschi et al., 2008] technique will be used.

The Combined WordSense Disambiguation Technique at a Glance

The current section presents a brief overview of the CWSD algorithm, as described in [Sorrentino, 2011].

The CWSD is a method for the automatic annotation of structured and semi-structured data sources that has been integrated in the MOMIS data integration system to aid the user in the lexical annotation of the data sources [Sorrentino, 2011]. CWSD relies on WordNet[10] as a lexical database and uses a combination of two different matching algorithms: a Structural Disambiguator

[10] `http://wordnet.princeton.edu/`

(SD) and a WordNet Domains Disambiguator (WND)[11] based on WordNet Domains[12]. The Structural Disambiguator disambiguates schema labels by using semantic relations derived from the structure of the data sources while WND relies on domain information obtained from WordNet Domains disambiguating schema labels.

In order to disambiguate the meaning of an ambiguous word, each WSD algorithm receives as input the *context* of the word. Many WSD algorithms present the context as a *bag-of-words*, i.e. a set of words that have to be disambiguated, and sometimes they insert in the context the information of the word position in the text. In CWSD, the context is composed by the set of labels (classes and attributes names) to be disambiguated, and the set of structural relations among these labels. The algorithm then computes a Common Thesaurus composed of a set of ODL_{I^3}[13] relations describing inter- and intra-schema knowledge among a set of data source schema in terms of four different kinds of relations: Synonym-of (SYN), Broader Term (BT), Narrower Term (NT), and Related Term (RT) [Sorrentino, 2011].

The Structural Disambiguator exploits the structural ODL_{I^3} in order to find corresponding lexical relations when an extensional relation holds between two terms. If there is a direct/chain of relations between two terms, semantically related meanings are identified in WordNet and the terms are annotated with these meanings.

The WND algorithm examines all possible synsets associated with a term and extracts the domains (as defined in WordNet Domains) connected to these synsets. Then, it computes the list of more frequent domains in the schema context. Finally, it compares the domain list with the domains associated with each term and it chooses all the synsets associated with the more frequent domains.

Semantic Mappings Validation

Starting from lexical annotations, semantic relations among attributes of the different Data Warehouse elements may be discovered by browsing the wide semantic network of WordNet [Bergamaschi et al., 2012]. In particular, the WordNet network includes[14]:

[11] WordSense Disambiguation is the task to computationally determine which sense of a word is activated bay its use in a particular context [Navigli, 2009].

[12] `http://wndomains.fbk.eu/`

[13] ODL_{I^3} is an object-oriented language (based on the *Object Description Language* (ODL)) with underlying Description Logic that has been used for information integration in the MOMIS data-integration project.

[14] WordNet includes other semantic and lexical relations such as antonym, cause etc. which are not relevant to the mapping discovery approach approach

Table 4.1 Coefficient Assignment

P_i/P_j	equi-level	roll-up	drill-down	related
same/synonyms	1	0.7	0.7	0.7
hypernyms	0.9	0.7	1	0.8
hyponyms	0.9	1	0.7	0.8
related terms	0.7	0.7	0.7	1
holonyms	0.7	1	0.3	0.8
meronyms	0.7	0.3	1	0.8

- **Synonym** relationships: defined between two labels annotated with the same WordNet meaning (*synset* in the WordNet terminology) (e.g., *client* is a synonym of *customer*)

- **Hypernym** relationships: defined between two labels where the meaning of the first is more general than the meaning of the second (e.g. *time period* is a hypernym of *year*). The opposite of *hypernym* is the *hyponym* relationship;

- **Meronym** relationships: defined between two labels where the meaning of the first is part of/member of the meaning of the second (e.g., *month* is a meronym of *year*). The opposite of *meronym* is the *holonym* relationship;

The *related terms* relation (that can be directly derived by WordNet) was added: two terms are related if they are connected by a hyponym or meronym relation to the same WordNet synset. Therefore, for each identified mapping, each label was first annotated by using the algorithm in [Sorrentino et al., 2010] and then the shortest path of semantic relations connecting the two elements into the WordNet network was computed. The goal was to validate the mappings by computing for each of them a similarity weight on the basis of the identified WordNet paths.

The similarity weights were computed by assigning every edge (i.e., WordNet relation) of the path a coefficient using the assignment rules in Table 4.1. The final similarity weight is given by multiplying the single coefficients (thus, long paths will have lower similarity weights than short or direct paths). These coefficients were defined by considering the type of WordNet relations and the type of mappings to be validated:

- an *equi-level* mapping is semantically similar to a *synonym* relation (coefficient equal to 1)

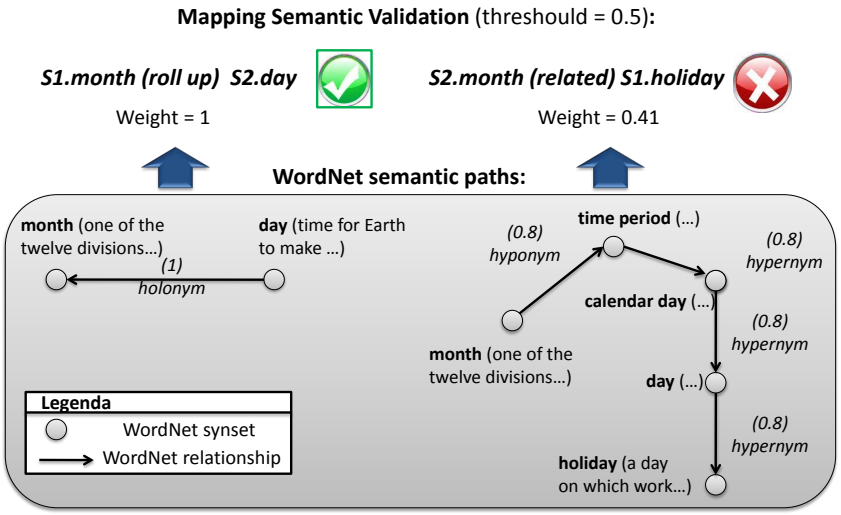


Figure 4.10: Semantic validation of the mappings

- a *roll-up/drill down* mapping is semantically similar to a *holonym/meronym* relation (coefficient equal to 1)
- a *related* mapping is semantically similar to a *related term* relation (coefficient equal to 1)

For the other mapping/relationship combinations coefficients (lower than 1) were assigned on the basis of their relevance. For example, to the combination *drill-down/holonym*, a low coefficient (0.3) was assigned as they semantically represent opposite concepts.

Consider the mappings $\omega_1 : S_1.month\ (roll-up)\ S_2.day$ and $\omega_2 : S_1.month\ (related)\ S2.holiday$ needed to be validated. The label *month* is annotated by CWSD with the meaning of "one of the twelve divisions of the calendar year", *day* as "time for Earth to make a complete rotation on its axis", and *holiday* as "a day on which work is suspended by law or custom". From the WordNet semantic network, a direct path was computed between *month* and *day* where the meanings are connected by a *holonym* relationship. Thus, by following the coefficients in Table 4.1, the mapping ω_1 was associated a similarity weight equal to 1. Between the terms *month* and *holiday*, a path of length 4 was discovered, and it was composed by *hyponym* and *hypernym* relationships (see Figure 4.10). Thus, a similarity weight equal to $0,41$ is computed and associated to ω_2 .

Finally, the mappings are validated by applying a threshold on their semantic weights. For example, by imposing a threshold of 0.5, the

mapping $S_1.month$ $(roll-up)$ $S_2.day$ is maintained, while the mapping $S_1.month$ $(related)$ $S_2.holiday$ is discarded.

4.2 Schema Integration

The semantic mappings offer a valid approach for representing semantic similarities as well as for integrating information from two or more heterogeneous DWs. However, any mapping-based DW integration methodology is far from providing a problem free solution. In fact, a correct set of mappings provides the possibility of integration parts of the DWs, which in some cases may not be a satisfying solution.

As an example, consider the DWs in Figures 1.1 & 1.2. As explained in Section 1.2, users may attempt to execute queries that are not compatible with all the DWs. This renders the mappings not highly efficient in real life scenarios where the number of DWs to be integrated can potentially be much higher (like in Business Intelligence Networks [Golfarelli et al., 2011], in virtual enterprises [Mowshowitz, 1986, Mowshowitz, 1997, Mowshowitz, 2003] or in CO-Opetition networks).

For the moment, the heterogeneity of the DW dimensions will not be analyzed from the perspective of the integration architecture, as the schema integration methodology that will be presented in this section is architecture-independent.

Assuming the data sources to be used for integration after a query has been written and executed depends on the query itself rather than on the integration architecture, then the user may find himself in one of the three following cases:

- the query is compatible with all the DWs of the network. In this cases, the result of the query is a combination of the result sets obtained from all the nodes of the network. This is the ideal case, and guarantees the *completeness* of the result;

- the query is compatible with a subset of the network. In this case, the result of the query may be the combination of the result sets of the queries executed on each individual data source. The main issue in this case is the completeness of the result and its interpretation. In some cases, a partial result may provide its utility, in other cases the impact may be negative. For example, the query “all the sales divided by trimester" is incorrectly formulated if the result is not obtained from combining the sales of all the nodes in the network. In case of incorrect results, the system may decide to inform or not to inform the user. Not informing the user is risky, as the user may be lead into thinking that the result is

complete. Informing the user may create confusion, as most of them are business oriented, and may not have a correct vision of the integration architecture and thus may not be able to draw a correct interpretation of the result obtained from the query.

- the query is compatible only with the local node where the query has been formulated. This is a restriction of the previous case, and renders the mappings and the integration effort useless.

In order to eliminate part of the limitations of the integration process, as discussed in the above cases, a proper solution is to perform schema integration on the data sources to be integrated. This is achieved by using the semantic mappings to import remote compatible dimension categories into a local schema, and subsequently populate it with consistent members. Integrating a dimension category in all/most of the similar dimensions of distinct DWs increases the compatibility of the DW schemas, and thus reduces the cases of incompatible/partial compatible queries. To the final user, the result of greater compatibility is higher integration and more complete results that can be obtained by writing more complex queries on the DW network [Olaru, 2012].

To better explain the schema integration methodology, a simple example showing how a dimension category may be imported from one dimension to another will be used. For this purpose, consider two dimensions

$$d_1 : (\mathrm{M}_1, <_1) \rightarrow (\mathrm{C}_1, \nearrow_1)$$

$$d_2 : (\mathrm{M}_2, <_2) \rightarrow (\mathrm{C}_2, \nearrow_2),$$

where $(\mathrm{M}_1, <_1)$ and $(\mathrm{C}_1, \nearrow_1)$ are the hierarchy domain and hierarchy schema of dimension d_1 (likewise, $(\mathrm{M}_2, <_1)$ and $(\mathrm{C}_2, \nearrow_2)$ for d_2). Let $\xi : \mathrm{C}_1 \rightarrow \mathrm{C}_2$ be a mapping among the two dimensions, discovered by using the approach in Section 4.1.

Let $c_i, c_j \in \mathrm{C}_1$ be such that $c_i \nearrow_1 c_j$ and $c'_i \in \mathrm{C}_2$. If $\xi(c_i) = c'_i$ and $\xi(c_j) \notin \mathrm{C}_2$, then d_2 is augmented with the category c_j[15] and with the roll-up relations derived from the semantic mappings. Thus, a new dimension $d_2^\sharp : (\mathrm{M}_2^\sharp, <_2^\sharp) \rightarrow (\mathrm{C}_2^\sharp, \nearrow_2^\sharp)$ is derived, where $\mathrm{C}_2^\sharp = \mathrm{C}_2 \cup \{c'_j\}$, $\nearrow_2^\sharp$ is extended to include the relations between c'_j and the categories of C_2. There are mainly two types of relations that extend $\nearrow_2^\sharp$:

- $c'_i \nearrow_2^\sharp c'_j$, $\forall\ c'_j$ that was imported in C_2;

[15] For the sake of clarity, we will refer to the category c_j imported in the dimension d_2 as c'_j.

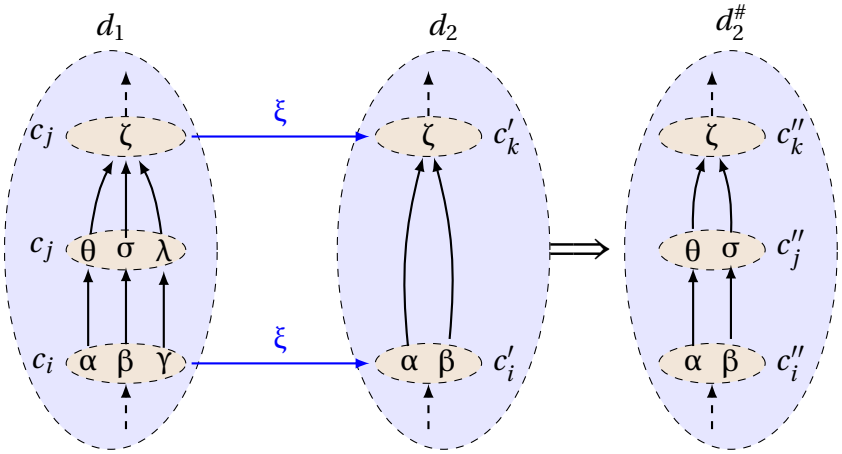


Figure 4.11: The *category* and *member* importation rule

- $\forall c_k \in C_1$ and $\forall c'_k \in C_2$, if $c_j \nearrow_1 c_k$ and $\xi(c_k) = c'_k$, then $c'_j \nearrow_2 c'_k$ (see Figure 4.11)[16].

The mapping ξ is extended to include the newly imported category. A new mapping $\xi^{\sharp} : C_1 \rightarrow C_2^{\sharp}$ is generated as follows:

$$\xi^{\sharp}(c) = \begin{cases} \xi(c), & \text{if } c \neq c_j \\ c_j, & \text{if } c = c_j \end{cases}$$

It is important to note that the importation of dimension categories maintains the partial order relations of the dimension schemas. The observation will be used when proving the coherency of the discovered mappings.

4.3 Instance Integration

The newly imported dimension categories may offer users the possibility of accessing views on data that depend on the imported category in an uniform way from the two or more dimensions that share the category. This possibility, however, is not granted, and depends on the members of the category that may be imported in the newly inserted dimension category.

For importing category members, the *d-chase* (dimension chase) procedure may be used, as defined in [Torlone, 2008]. The *d-chase* approach is inspired by the *chase* algorithm introduced in [Abiteboul et al., 1995] for reasoning on dependencies in database theory. The *d-chase* procedure involves creating a

[16]The second case is conceptually similar to the complex mapping generation Rule 6.

tableau T over $C_2 \cup \xi(C_2)$[17][18] composed of tuples having the values occurring in the dimension instances, or *variables* that will later be changed to constant values. The procedure is based on the recursive application of a *chase step*, that for every roll-up relation $c_i \nearrow_2 c_j$, and $\forall t_1, t_2 \in T$, if $t_1[c_i] = t_2[c_i]$, then it must be that $t_1[c_j] = t_2[c_j]$. If $t_2[c_j]$ is a variable, then it is *promoted* to the constant value $t_1[c_j]$. This implies that $t_1[c_j] \in M_2^{\#}$. Furthermore, it must be that $t_1[c_j] <_2^{\#} t_2[c_j]$.

If $t_1[c_j] \neq t_2[c_j]$ and $t_2[c_j]$ is not a variable, then T is inconsistent, as the two values are clashing. This is an indication that the two dimensions are inconsistent, and a resolutive approach must be used, as will be shown in Section 4.5.

A graphical visualization of the category and member importation step is presented in Figure 4.11, where d_1 and d_2 represent two dimensions, and ξ a mapping among them. Following the category and member importation step, the category c_j is imported in the dimension d_2 (becoming thus dimension $d_2^{\#}$) and is populated only with values θ and σ, as those are the only relevant values for the second dimension.

Table 4.2 in Section 4.4 provides an example Tableau used for integration.

4.4 Dimension Integration Discussion

Independently of the used formalism, it is clear that the mappings and the dimension instances play major role in the integration process. There are at least two important aspects that must be discussed.

First of all, the *correctness* of the dimension mappings are crucial for guaranteeing correct results when performing query rewriting and integration, as incorrect mappings may lead to integrating distinct and incoherent data, or integrating the correct data in an incoherent way.

Secondly, the completeness and the correctness of the instance increase the chances of correctly computing the dimension mappings (as highlighted in Section 4.1.2) and allow to correctly import category members from one dimension to another.

Furthermore, uniforming dimensions may increase the analysis opportunities of users accessing only the individual DWs. Apart from increasing query compatibility when querying multiple DWs simultaneously, it offers users of each individual DW increased querying capabilities. Indeed, considering the example in the previous section, a user of DW_2 is offered, after the integration

[17] $\xi(C_2)$ denotes the members of $c_j \in C_2$ for which $\exists c_i \in C_1$ such that $\xi(c_i) = c_j$

[18] If $\xi : C_1 \rightarrow C_2$, then $\xi^{-1} : C_2 \rightarrow C_1$ also maps categories of C_2 to categories of C_1 (the mapping is $1:1$). With a little abuse of notation, we will also refer to ξ^{-1} as ξ.

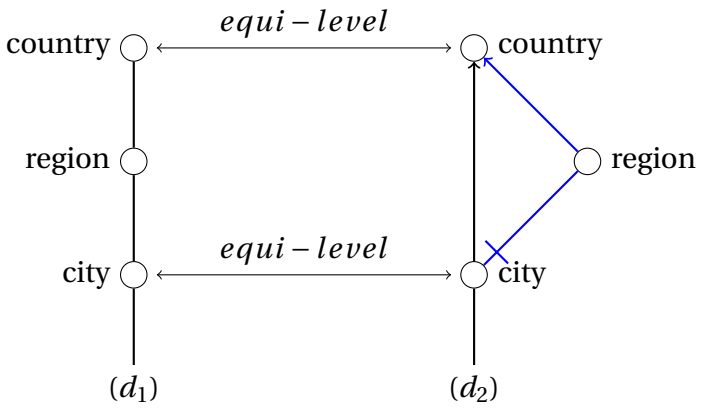


Figure 4.12: Integration using DFM

process, the possibility of querying the local DW by using queries involving also the dimension category c'_j, that was not initially contained in dimension d_2.

This is the case in some integration architectures (not all), where the integration is being made by intervening locally, on each single DW. For example, in a network of CO-Opetiting businesses, developers may decide to uniform the dimensions of analysis, and maintain a distributed-network approach to information exchange.

As an example, consider the dimensions in Figure 4.12, represented using the *Dimensional Fact Model* (DFM) as the formalism for the integration process, as presented in [Olaru and Vincini, 2012]. Considering that the two dimensions, d_1 and d_2 belong to two distinct DFM schemas f_1 and f_2, and that $region \notin A_2$[19], and supposing that the mapping discovery procedure proposes (as discussed in Section 4.1.3) the mappings $\omega_1 : d_1.city < equi-level > d_2.city$ and $\omega_2 : d_1.country < equi-level > d_2.country$, then the schema integration step introduces the *dimensional* attribute $region$ to d_2, as highlighted in Figure 4.12.

Note that the imported attribute ($region$) is optional, as it may be that not all $cities$ have a $region$ (it depends on the instances of d_1 and d_2, as will be shown later). However, users accessing DW_2 can now compute aggregated information grouped by region, for some/all regions of the instance.

Suppose Table 4.2 represents the tableau of the *d-chase* procedure, where the first three tuples belong to dimension d_1, while the last three tuples belong to dimension d_2. By recursively aplying the chase procedure, then the following variable assignments are made:

[19] A_1 and A_2 are the dimensional attributes sets of schemas f_1 and f_2.

- v_1 =ER
- v_2 =TU

Variable v_3 is assigned no constant value, thus the dimensional attribute $region$ remains optional, as there are cities (ROME) for which no correct region could be computed by using the instances of the two dimensions. Had all the variables been assigned a constant value, then the dimensional attribute $region$ would have been modified into a mandatory dimensional attribute ($region \in A_2$).

Note that the integration process, apart from uniforming the querying capabilities of users accessing the two DWs, offers furthermore users of DW2 the possibility of aggregating information by additional dimensional attributes (in the previous example, by the attribute "region" which was not initially contained in dimension d_2).

Table 4.2 Tableau

city	region	country	dimension
MODENA	ER	ITALY	d1
FLORENCE	TU	ITALY	d1
BOLOGNA	ER	ITALY	d1
MODENA	$v1$	ITALY	d2
FLORENCE	$v2$	ITALY	d2
ROME	$v3$	ITALY	d2

4.5 Instance Integration Resolution

The *d-chase* approach proposes a direct equality of the dimension members (or values of the dimensional attributes, as called in the DFM) which is adeguate when identical values are used for representing the same, real life concept.

Unfortunately, most of the times, different working groups use different values even for identifying the same concepts, and this may be due, for example, by the use of abbreviations (e.g., "Emilia Romagna" vs. "E. Romagna"), use of different languages (e.g., "Turin" [eng.] vs. "Torino" [ita.]), different formats (e.g., weekday names vs. weekday numbers), or simply by human error. As there is no general method of checking inconsistent values (a part from using a standard reference dictionary), it is safer to use resolutive functions for assigning the members of a category. The use of resolutive functions has at least two benefits:

- correctly assign members to a category: although this may seem trivial, it is not always guaranteed when using a simple equality approach (like the *d-chase*) where different values for the same concept lead to the interpretation that the concepts are different;

- correction of wrong values: resolution functions may be used, as will be illustrated further, to correct incorrect dimension members; as the mapping signifies a semantic similarity of the categories, then the values must also share some similarity as they cannot be completely distinct. In the approach that will be highlighted in the following, this similarity will be enforced by using a clustering technique.

For resolving instance level inconsistencies, the *RELEVANT* clustering approach will be used. Rather than enforcing a direct value approach, a *class* similarity will be used, where the class is represented by the cluster of values, as computed by *RELEVANT*.

4.5.1 *RELEVANT* at a Glance

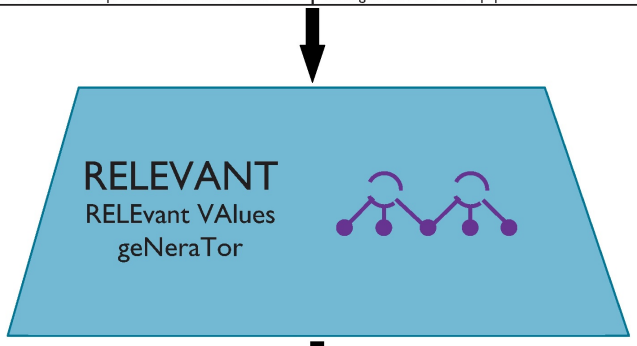
The current section provides a brief description of the *RELEVANT* clustering approach, as described by the authors in [Bergamaschi et al., 2007e, Bergamaschi et al., 2007b]. The main idea behind *RELEVANT* is to extract a new kind of metadata (*relevant values*) that represent an attribute domain with a synthetic description.

The approach was developed inside the MOMIS data integration system [Bergamaschi et al., 2001, Beneventano et al., 2000]. In MOMIS, data is represented by means of a Global Virtual View (GVV) composed of global attributes (GAs). *RELEVANT* extracts representative values for the GAs, that are described by a name (the label of the cluster) and a set of attribute domain values associated with the relevant value name. The motivation of the approach is that by analyzing an attribute's domain, strongly related values that are clustered may be found.

Figure 4.13 provides an example of clusters computed from sets of values. For example, the values *adhesive agents*, *adhesive labels*, *adhesive tapes* and *adhesives* are clustered, and the relevant name "adhesives" is elicited from the set of values as the relevant value of the set.

RELEVANT performs clustering by building a binary representation of the attribute values and exploits three different kinds of measures to build some structure on the flat set of binary representations: *syntactic similarity*, *dominance measure* and *lexical similarity*.

Adhesive agents	Finished and semifinished product tests
Adhesive labels	Finished and semifinished product weight tests
Adhesive tapes	Injection
Adhesives	Injection blow moulding
Aluminum and magnesium casting	Injection moulded parts
Assembling operations	Injection moulding machines, general purpose
Assembly	Intrusion moulding
Assembly of printed circuits	Mould engineering
Assembly/modules	Moulded parts
Automatic assembly machines	Moulded parts from semifinished products
Blake assembly	Moulding
Blow moulding	Moulds for ceramics
Blow moulding machines	Moulds for plastic and nonmetal parts
Blowing	Moulds manufacturing
Blowing and injection moulding or forming	Moulds, dies
Cast iron casting	Normal moulding
Cast moulded parts	Other finished and semifinished product tests
Casting	Physical tests
Casting dies	Plastic castings
Casting machines for open moulds	Plastic technical products moulding
Casting with other moulding methods	Plastic molding
Design	Precision moulds for thermoplastic resins
Design/development	Rotation moulding
Designing	Sandwich moulding
Dip moulding	Test modules
Family mould injection	Testing
Finished and semifinished product dimensional tests	Testing of modules and equipment



Design	Design; Design/development
Adhesives	Adhesive agents; Adhesive labels; Adhesive tapes; Adhesives
Assembly	Assembling operations; Assembly; Assembly of printed circuits; Assembly/modules; Automatic assembly machines; Blake assembly
Testing	Finished and semifinished product dimensional tests; Finished and semifinished product tests; Finished and semifinished product weight tests; Physical tests; Test modules; Testing; Testing of modules and equipment
Plastics molding	Plastics Molding
Moulding	Blow moulding; Blow moulding machines; Cast moulded parts; Casting with other moulding methods; Family mould injection; Injection; Injection blow moulding; Injection moulded parts; Injection moulding machines, general purpose; Intrusion moulding; Mould engineering; Moulded parts; Moulded parts from semifinished products; Moulding; Moulds for Ceramics; Moulds manufacturing; Moulds, Dies; Normal moulding; Precision moulds for thermoplastic resins; Rotation moulding; Sandwich moulding
Casting	Aluminum and magnesium casting; Cast iron casting; Cast moulded parts; Casting; Casting Dies; Casting machines for open moulds; Casting with other moulding methods; Plastic castings
Injection	Blowing and injection moulding or forming; Family mould injection; Injection; Injection blow moulding; Injection moulded parts
Moulding; blowing	Blow moulding; Blow moulding machines; Blowing; Cast moulded parts; Casting with other moulding methods; Dip moulding; Family mould injection; Injection blow moulding; Injection moulded parts; Intrusion moulding; Mould engineering; Moulded parts; Moulded parts from semifinished products; Moulding; Moulds for Ceramics; Moulds manufacturing; Moulds for plastic and non-metal parts; Moulds, Dies; Normal moulding; Plastic technical products moulding; Precision moulds for thermoplastic resins; Rotation moulding; Sandwich moulding

Figure 4.13: Example of relevant values

Syntactic Similarity is based on the idea that terms referring to the same/similar object(s) can have the same etymology and share a common root. Similarity measures (e.g., Levenshtein distance; see [Frakes and Baeza-Yates, 1992]) may thus be used to compute how "close" two values are. The attribute value's words are mapped in an abstract space, then a syntactic similarity function is use to compute the edit distance between two different values.

Dominance Measure is a term used to state that an attribute a_1 has a more "general" meaning than a term a_2. Following this idea, *RELEVANT* extracts a dominance relationship between attribute values that is used for generating the value synthesizing the cluster of attribute values.

The intuition behind using the dominance measure is that in database instances it is frequent for string domains to have values composed by many words and abbreviations; also the same word or group of words can be further specialized in different ways. For example, the attribute describing a kind of production for a mechanical enterprise could contain the value "mold" and the values "mold ejectors", "mold engineering", etc. Dominance may be built either by syntactic properties (e.g., if the string X contains the string Y, then Y dominates X as it is more general), or by semantic dominance (e.g., "sedan" is dominated by "car, that has a broader semantic meaning). In *RELEVANT*, the dominance is considered as a *partial order* relation among values that describes an oriented graph. *RELEVANT* then attempts to build clusters around outgoing edges of the graph (nodes representing values that dominate large sets of values).

Lexical Similarity is used by exploiting the WordNet lexical database. The disambiguation of a term by using WordNet is done by assigning a synset to every term (or attribute value) in the database. *RELEVANT* tries to exploit the knowledge that two terms sharing identical/similar synsets should belong in the same cluster, even though they have different values. The algorithm thus attempts to cluster terms that have similar synsets, rather than terms that are assigned completely distinct synsets.

RELEVANT uses a combination of these three measures, and for each category, computes a series of clusters, which are pairs of elements $rv =< rvn, values >$, where rvn is the name (or label) of the cluster (also one of the values of the clusters), and $values$ is the set of values contained in the cluster.

4.5.2 Importing Values with *RELEVANT*

RELEVANT performs clustering on attribute values that are syntactically and semantically correlated; it is thus safe to assume that identical or similar values

representing the same real-life concept should belong to the same cluster. Following this observation, one possible way of solving instance inconsistency is to *enforce* that values that belong to the same cluster roll-up to values that are also clustered by *RELEVANT* [Guerra et al., 2012].

For start, the condition will be analyzed on a single dimension. This serves only as an example for illustrating the clustering condition that will later be adapted for two or more dimensions.

Let c be a category of some dimension $d : (\mathrm{H}, \nearrow) \rightarrow (\mathrm{M}, <)$, and m be a member of that category ($m \in \mathfrak{m}(c)$). Let $\mathfrak{R}_d(m)$ be a function, that given the dimension d, assigns each member its relevant cluster $rv_m =< rvn_m, values_m >$. Note that the function is in most cases not injective, as clusters are usually formed by multiple values.

Given any dimension d, and $c_i, c_j \in \mathrm{C}$ such that $c_i \nearrow c_j$, then by enforcing that *RELEVANT* conditions hold on the dimension, it must be that

$\forall m_{i_1}, m_{i_2} \in \mathfrak{m}(c_i)$ such that $\mathfrak{R}_d(m_{i_1}) = \mathfrak{R}_d(m_{i_2})$, then
  if $\exists\ m_{j_1}, m_{j_2} \in \mathfrak{m}(c_j) :\ m_{i_1} < m_{j_1}$ and $m_{i_2} < m_{j_2}$, then
    $\mathfrak{R}_d(m_{j_1}) = \mathfrak{R}_d(m_{j_2})$.

The same condition may be applied to n dimensions, say $d_1, d_2, \ldots, d_n$ that contain pairs of categories that are mapped.

For better understanding this concept, it is safe to assume that a mapping ξ is reflexive and transitive. Thus, given two dimensions d_i and d_j, and $c_i \in \mathrm{C}_i$ and $c_j \in \mathrm{C}_j$, if $\xi_{i,j}(c_i) = (c_j)$, then $\xi_{j,i}(c_j) = (c_i)$[20]. Furthermore, if $c_k \in \mathrm{C}_k$ (the category set of some dimension d_k), and $\xi_{j,k}(c_j) = (c_k)$, then it must be that $\xi_{i,k}(c_i) = (c_k)$[21]. Although these properties are not always guaranteed (it depends on the mapping generation methodology), for a correct and complete mapping set among n dimensions it is safe to assume that the two properties are always guaranteed.

In a context of n dimensions, the clustering rule may be enforced on sets of mapped categories. Assuming that $c_{i_1}, c_{j_1} \in \mathrm{C}_1; \ldots; c_{i_n}, c_{j_n} \in \mathrm{C}_n$ are categories such as $c_{i_1} \nearrow c_{j_1}; \ldots; c_{i_n} \nearrow c_{j_n}$ and $\xi_{x,y}(c_{i_x}) = c_{i_y}$, and $\xi_{x,y}(c_{j_x}) = c_{j_y}$ $\forall x, y \in \{1, 2, \ldots, n\}$, then the clustering rule may be enforced on the union of the member sets of the categories:
$\forall m_{i_1}, m_{i_2} \in \mathrm{M}_1 \cup \mathrm{M}_2 \cup \cdots \cup \mathrm{M}_n$, and $\forall m_{j_1}, m_{j_2} \in \mathrm{M}_1 \cup \mathrm{M}_2 \cup \cdots \cup \mathrm{M}_n$
  if $\exists x, y$ such that $m_{i_1} <_x m_{j_1}$ and $m_{i_2} <_y m_{j_2}$, and $\mathfrak{R}(m_{i_1}) = \mathfrak{R}(m_{i_2})$, then
    $\mathfrak{R}(m_{j_1}) = \mathfrak{R}(m_{j_2})$
  Note that the condition is identical to the case of one dimension, but for the

[20] Assuming that $\xi_{i,j} : \mathrm{C}_i \rightarrow \mathrm{C}_j$ is a function that maps categories of C_i to categories of C_j and $xi_{j,i} : \mathrm{C}_j \rightarrow \mathrm{C}_i$ maps categories of c_j to categories of C_i.

[21] Assuming that a mapping between dimensions d_j and d_k has been defined

fact that the *RELEVANT* clusters are computed over the union of the member sets of all mapped dimension categories, rather than the individual member sets of each category. Such approach compensates the different syntactic and semantic values used to represent the same concept.

Example. Consider a scenario of four dimensions, three of which having the dimension categories $city$ and $region$ (even with different labels), that are mapped together, and the fourth containing only the category $city$ (and possibly others that are not relevant for this example). Suppose that by using the mappings, the category $region$ was imported in the forth dimension and it must be populated with members. Suppose the members of the categories are as in Table 4.3 (one color for each dimension), where the members of the category $region$ of the forth dimension are all *null* values.

Analyzing the clusters of the members and the type of relation to impose (as discussed previously), one may find correct clusters of category members that roll-up to members of the same cluster, like {Modena | Modena} → {Em.Romagna | E.Romagna}. The *null* values may be replaced with any value of the *RELEVANT* cluster, or by the label of the cluster. For example, the $region$ of the $city$ "Rome" in the fourth dimension (red values) is "Lazio", as the clusters computed by *RELEVANT* are {Roma | Rome | Rome} → {Lazio | Lazio | ~~*null*~~}.

Table 4.3 Category members

city	region		city	region
Roma	Lazio		Florence	Tuscany
Firenze	Toscana		Bologna	Emilia Romagna
Modena	Em.Romagna		Rome	Lazio
Milano	Lombardia		Turin	Piedmont
Palermo	Sicilia		Palermo	Piedmont
Palermo	Sicilia		Rome	*null*
Milano	Lombardia		Torino	*null*
Pisa	Toscana		Bologna	*null*
Modena	E.Romagna		Palermo	*null*
Palermo	Sicilia		Firenze	*null*

Incorrect values are identified as those values that don't maintain the roll-up clustering relation. For example, the roll-up clustering rule enforces that {Palermo | Palermo | Palermo | Palermo} → {Sicilia | Sicilia | ~~Piedmont~~ | ~~*null*~~}. The value {Piedmont} is detected as inconsistent, because it doesn't belong in the cluster formed by the values {Sicilia | Sicilia}. In the example, the values of the regions contained in the dimension instances are grouped in two distinct

clusters. To choose the *correct* value to which the member {Palermo} rolls-up, a simple voting mechanism may be used. The cluster {Sicilia | Sicilia} is formed by two values (the value is encountered twice in the instances), while the cluster {Piedmont} is formed by one value. The cluster of the majority (being composed of more values) is thus considered as the *correct* one. By enforcing this rule, the city {Palermo} must roll-up to the relevant value of the cluster region, in this case either of the two values {Sicilia | Sicilia}. In presence of more than one cluster, the *null* value may be replaced by the label of the dominant cluster.

4.6 Integration Architectures

One positive aspect of the dimension mappings and integration approach is that it may be used under different integration architectures, which offers flexibility both to developers as to end users.

One possible application is *Peer-to-Peer Data Warehousing* (for example in the context of the *Business Intelligence Network*or BIN [Golfarelli et al., 2011]), where users have the possibility of choosing at query time whether to use all the DWs of the network, or just a subset of the peers. In this approach, mappings have to be computed between every pair of nodes that want to exchange information. Given the high number of mapping sets (see Figure 4.14), a manual integration approach may prove inefficient, especially in complex scenarios with a high number of peers. That is why an automatic approach could ease the task of developing the mapping-based infrastructure for allowing the actual integration process. After computing the mappings, query rewriting mechanisms (like the one presented in [Golfarelli et al., 2010]) may be used for integration purposes.

One important note about this architecture is that no global schema or global DW has to be computed, which means that there is no point of reference

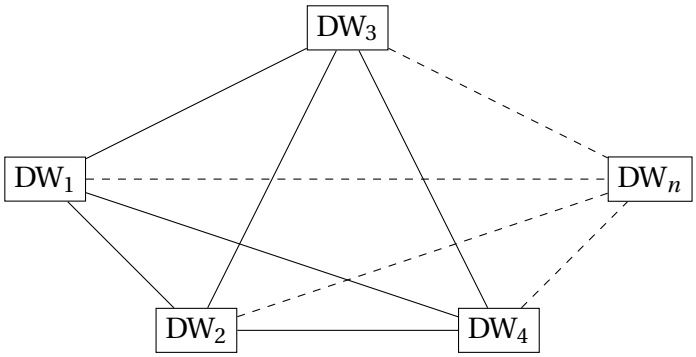


Figure 4.14: Peer-to-Peer Data Warehouse

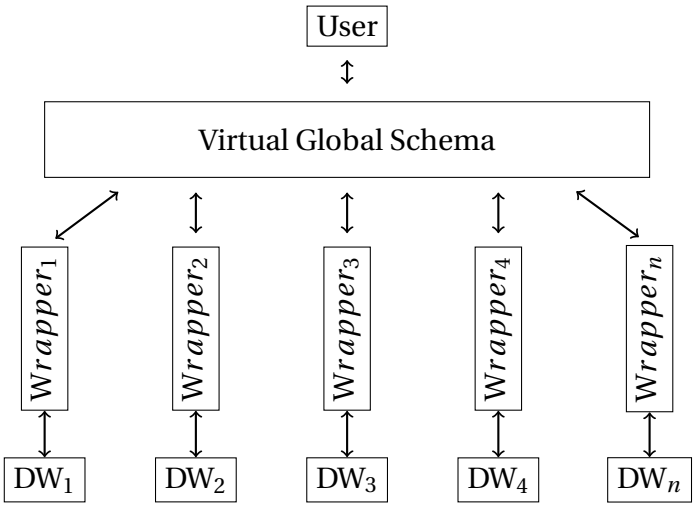


Figure 4.15: Federation Data Warehouse

for the union of all the DWs. It may happen thus that when each peer computes the same query over the entire network, the result is replicated n times over the network, with negative results to data redundancy and network load.

A different approach is the Federation of DWs, as discussed in Section 2.5, where the architecture provides a single, unified view of the available DWs. The integration in this case is virtual, as no global DW is physically built. Rather, a unified global schema describing the union of the individual schemas is computed. Users accessing the FDB write queries on the virtual global schema, the query is then rewritten by using wrappers (see Figure 4.15) that translate the query from the global schema to the local schema. Information is then returned by the wrapper to the system that assembles the local results from the individual DWs.

In the following, a simple, iterative approach for building the global virtual schema is presented.

Consider a scenario of n distinct, mapped DW dimensions $d_1, d_2, \ldots, d_n$, where the i-th dimension is defined as $d_i : (\mathrm{M}_i, <_i) \rightarrow (\mathrm{C}_i, \nearrow_i)$. The mapping between d_i and d_j will be denoted by $\xi_{i \otimes j}$. The symbol $d_{i \otimes j}$ will denote the dimension d_i after importing the compatible dimension categories from dimension d_j to dimension d_i. Note that $d_{i \otimes j} \neq d_{j \otimes i}$, the difference between the two dimensions being caused by the distinct (unmapped) dimension categories. A dimension mapping (denoted by $\xi_{\{i \otimes j\} \otimes k}$) may be computed between $d_{i \otimes j}$ and d_k as follows[22][23]:

[22] $\mathrm{C}_{i \otimes j}$ indicates the category set of dimension $d_{i \otimes j}$

[23] It is assumed that $\xi_i(c) = \xi_j(c)$, if $c \in \mathrm{C}_i \cap \mathrm{C}_j$

$\xi_{\{i \otimes j\} \otimes k} : \mathrm{C}_{i \otimes j} \rightarrow \mathrm{C}_k$, where

$$\xi_{\{i \otimes j\} \otimes k}(c) = \begin{cases} \xi_i(c) & \text{if } c \in \mathrm{C}_i \\ \xi_j(c) & \text{if } c \in \mathrm{C}_j. \end{cases}$$

After these considerations, a global dimension schema may be computed by iteratively integrating the dimensions into one global dimension. For this purpose, Algorithm 2 starts from dimension d_1, and subsequently builds into the global dimension d_t the other dimensions.

Algorithm 2 Global Schema Computation

```
1:  d_T := null
2:  CONTINUE = FALSE
3:  i:=1
4:  repeat
5:      d_T = d_i
6:      for j := 1 to n; j ≠ i do
7:          if ∃c ∈ C_T, c ≠ C_{base_T} : ξ_{Ti}(c) = c_{base_i} then
8:              CONTINUE = TRUE
9:              EXIT FOR
10:         else d_T := d_t ⊗ d_i
11:         end if
12:     end for
13:     i:=i+1
14: until CONTINUE
```

Note that the Algorithm may start from a different dimension, thus yielding a possible different global schema for the global dimension. This is the reason why the algorithm considers the exit condition at line 7, and starts again from dimension d_{i+1}.

When integrating two dimensions d_i and d_j, depending on the direction of the integration, the dimensions $d_{i \otimes j}$ and $d_{j \otimes i}$ may be different. For the sake of clarity, consider the example in Figure 4.16 where two dimensions may be integrated by using the mapping $\xi_{j,i}$ that maps only the dimension category c_l to the dimension category c_i of dimension d_i. The resulting dimension $d_{j \otimes i}$ is shown on the right. Conversely, considering that the inverse mapping $\xi_{i,j}$ would map only the category c_i to the category c_l, then a resulting dimension $d_{i \otimes j}$ would be identical to the dimension $d_{j \otimes i}$, but for its base category c_k.

The main reason for the difference is that the schema integration procedure imports categories from a certain category upwards. When two dimensions have different bottom categories, the result of the integration is different

if performed from one direction to another, or vice-versa. When a mapping $\xi_{i,j}$ (or $\xi_{j,i}$) maps the bottom categories of two dimensions d_i and d_j, then the integration process yields identical dimensions $d_{i\otimes j}$ and $d_{j\otimes i}$.

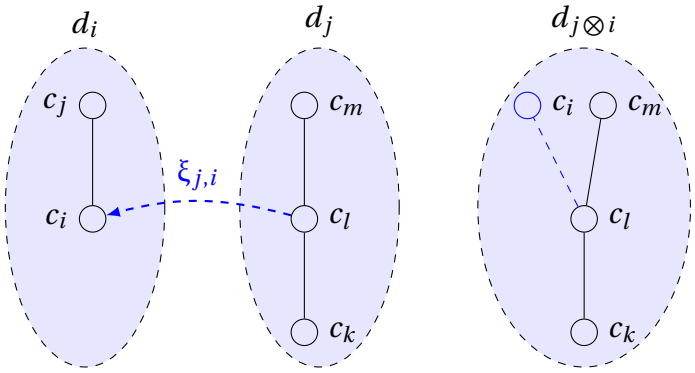


Figure 4.16: Example integration differences

Chapter 5

Dimension Mappings Properties

The semantic mappings of dimension identify similar categories used to aggregate information at a certain level of abstraction. Differently from flat schemas, where attributes may be mapped and combined independently, heterogeneous DWs must be mapped by maintaining the partial order structure that the dimensions have. This is required in order to not alter the view of the information that analysts have of their own business.

When generating semantic mappings among DW elements, it is not always trivial to maintain the structure of the dimension hierarchies, as elements are mapped individually. A dimension mapping quality analysis is thus required to ensure that the analytical capabilities are not altered when heterogeneous DWs are mapped and integrated.

For this purpose, three dimension mappings quality properties, namely *coherency*, *soundness* and *consistency*, defined in [Cabibbo and Torlone, 2005, Torlone, 2008] will be used.

For the following definitions, assume d_1 and d_2 are two dimensions defined as $d_1 : (\mathrm{M}_1, <_1) \rightarrow (\mathrm{C}_1, \nearrow_1)$ and $d_2 : (\mathrm{M}_2, <_2) \rightarrow (\mathrm{C}_2, \nearrow_2)$. Furthermore, $\nearrow_1^*$ and $\nearrow_2^*$ are the reflexive and transitive closures of the roll-up relations $\nearrow_1$ and $\nearrow_2$. Assume μ is a generic mapping $\mu : \mathrm{C}_1 \rightarrow \mathrm{C}_2$ that maps some categories of d_1 to categories of D_2.

Coherency is a property concerning the dimension schemas, in particular the partial order relation imposed on the category set. The mapping μ is coherent if $c_i \nearrow_1^* c_j \Leftrightarrow \mu(c_i) \nearrow_2^* \mu(c_j)$. The coherency property states that the roll-up relations among attributes are maintained through the mapping between the dimensions. That is, there are no "cross mappings", like highlighted in Figure 5.1, where the category i_1 of the dimension d_1 has been mapped to category j_2 of dimension d_2. This is to state that i_1 and j_2 express the facts under analysis at the same granularity (or aggregation level). If i_2 would also be

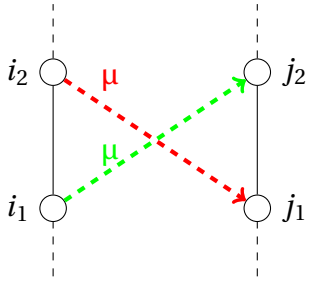


Figure 5.1: Incoherent mapping

mapped to j_1. This would state that i_2, which is more aggregated than i_1 (there is a $1:n$ relation among the members of the two categories) has the same granularity of j_1, which is more detailed (lower granularity) than j_2 (which is the same as i_1). This is, of course, a non-sense mapping.

Soundness is a property concerning the members of categories, in particular a mapping is sound if the member sets of mapped categories are identical. That is, $\mathfrak{m}(c) = \mathfrak{m}(\mu(c))$ for all $c \in C_1$. Although the property strongly depends on the DW instances, and although not strictly required for correct dimension integration, it is useful for guaranteeing other properties, like *homogeneity*.

Consistency ensures that the roll-up function between members of mapped categories is maintained. Formally, $\forall m_{i_1}, m_{j_1} \in M_1$ such that $m_{i_1} <_1^* m_{j_1}$ then $\exists m_{i_2} \in \mathfrak{m}(\mu(d(m_{i_1})))$ and $m_{j_2} \in \mathfrak{m}(\mu(d(m_{j_1})))$ such that $m_{i_2} <_2^* m_{j_2}$ (where $<_1^*$ and $<_2^*$ are the transitive closures of $<_1$ and $<_2$).

In other words, if two members roll-up in one dimension, then the mapping should guarantee that there are two equivalent members in the other dimension that maintain the roll-up relation.

A mapping that is coherent, sound and consistent is called a *perfect mapping*.

5.1 Mapping Properties Analysis

In this section, the dimension mapping properties will be analyzed under the mapping methodology in Section 4. Notably, the mapping generation step guarantees coherency, meanwhile the importation step preserves soundness and consistency and in some cases may render a mapping that is neither sound nor consistent into a mapping that is sound and/or consistent.

For the rest of the section assume that $d_1 : (M_1, <_1) \rightarrow (C_1, \nearrow_1)$ and $d_2 : (M_2, <_2) \rightarrow (C_2, \nearrow_2)$ are tho dimensions belonging to some Data Warehouses

DW_1 and DW_2, and that $\xi : C_1 \rightarrow C_2$ is a mapping generated by the methodology presented in Section 4.

5.1.1 Coherency

One important result is that *coherency* is unconditionally guaranteed.

Theorem 1. *The mapping* ξ *is coherent.*

The theorem may be proved by using the graph isomorphisms that in a directed graph preserve node links (and paths) and their order.

Proof. The label of the directed graphs expressing the dimension hierarchy of a generic dimension $d : (M_1, <) \rightarrow (C, \nearrow)$ may be expressed by means of the function defined as:

$$f : \nearrow^* \rightarrow \mathbb{Q}^+$$
$$f\left[(c_i, c_j)\right] = \frac{\#\mathfrak{m}(c_i)}{\#\mathfrak{m}(c_j)}$$

where $\#\mathfrak{m}(c_i)$ is the cardinality of the member set of category c_i (likewise for c_j). Let $\mathbb{G}_{\mathbb{1}} = \left[(C_1, \nearrow_1), f_1\right]$ and $\mathbb{G} = \left[(C_2, \nearrow_2), f_2\right]$ be the directed labeled graphs representing the dimension hierarchies of dimensions d_1 and d_2, and let $\mathbb{G} = \left[(C, \nearrow), f\right]$ be the graph that is *subgraph*-isomorphic to both G_1 and G_2. Considering the subgraphs of the mapped categories, it must be that $\exists\, C_{T_1} \subseteq C_1, \exists C_{T_2} \subseteq C_2$ such that $G_{1_{|C_{T_1}}} = \left[(C_{T_1}, \nearrow_{1_{|C_{T_1}}}), f_{1_{|C_{T_1}}}\right]$ and $G_{2_{|C_{T_2}}} = \left[(C_{T_2}, \nearrow_{2_{|C_{T_2}}}), f_{2_{|C_{T_2}}}\right]$ are isomorphic to $\mathbb{G}$, where $G_{1_{|C_{T_1}}}$ and $G_{2_{|C_{T_2}}}$ are the restrictions of G_1 and G_2 to C_{T_1} and C_{T_2} respectively. It follows that there are two graph isomorphisms $w_1 : G_{1_{|C_{T_1}}} \rightarrow \mathbb{G}$ and $w_2 : G_{2_{|C_{T_2}}} \rightarrow \mathbb{G}$. Let $w = w_2^{-1} \circ w_1$; w is also a graph isomorphism from $G_{1_{|C_{T_1}}}$ to $G_{2_{|C_{T_2}}}$.

The graph isomorphism ensures that for all c_i and $c_j \in C_{T_1}$ it stands that $c_i \nearrow_1^* c_j \Rightarrow w(c_i) \nearrow_2^* w(c_j)$. Let $\xi_{|C_{T_1}}$ be the restriction of ξ to C_{T_1}. By construction, $\xi_{|C_{T_1}} \equiv w$. It follows that $c_i \nearrow_1^* c_j \Rightarrow \xi_{|C_{T_1}}(c_i) \nearrow_2^* \xi_{|C_{T_1}}(c_j)$.

Thus, $\xi_{|C_{T_1}}$ is coherent. By extension, ξ is also *coherent.* □

5.1.2 Soundness and Consistency

Although the first step of the integration methodology produces a coherent mapping, soundness and consistency are guaranteed only in certain cases. In order to verify whether soundness is verified, two steps must be performed. First, the initial mapping must be checked. Formally, for all categories $c \in C_1$,

it must be that $\mathfrak{m}(c) = \mathfrak{m}(\xi(c))$. This is a simple inclusion test that will be analyzed no further. Secondly, the soundness and consistency property must be verified after performing the category and member importation. For this purpose, consider $\xi^{\#}$ as the mapping between the dimensions after the category and member importation step.

The following theorem provides a sufficient condition for guaranteeing soundness and consistency when importing categories and members.

Theorem 2. *If ξ is sound and consistent, then $\xi^{\sharp}$ is also sound and consistent.*

Proof. If ξ is sound, then $\mathfrak{m}(c_i) = \mathfrak{m}(c_k)$. The member importation rule states that if $c_i \nearrow_1 c_j$ and $c_k \nearrow_2^{\sharp} c'_j$ and $\xi(c_i) = c_k$, then for all $m_i \in \mathfrak{m}(c_i), m_j \in \mathfrak{m}(c_j)$ and $m_k \in \mathfrak{m}(c_k)$ such that $m_i <_1 m_j$ and $m_i = m_k$, then it must be that:

(a) $m_j \in \mathfrak{m}(c'_j)$;

(b) $m_k <_2^{\sharp} m_j$ (see Figure 4.11).

If ξ is sound, from (a) follows that $\xi^{\sharp}$ is also sound.
If ξ is consistent, from (b) follows that $\xi^{\sharp}$ is also consistent. □

Theoreme 2 provides a *sufficient*, but not *necessary* condition for soundness and consistency. In fact, there may be cases when mapping two dimensions where the initial mapping ξ is neither sound nor consistent, but the final mapping $\xi^{\sharp}$ becomes sound and/or consistent after the second step of the integration methodology. For example, Figure 5.2 provides two dimensions and a mapping ξ that is neither sound nor consistent, as $\mathfrak{m}(c_j) \neq \mathfrak{m}(c_l)$ (the member σ belongs to $\mathfrak{m}(c_j)$ but not to $\mathfrak{m}(c_l)$) and $\rho^{c_i \to c_j} \neq \rho^{c_k \to c_l}$ (member β rolls-up to member α in dimension d_1, but rolls-up to no member of c_l in dimension d_2). Note that dimension d_2 is heterogeneous. Assuming the methodology generated the mapping ξ (see Figure 5.2), step 2 of the methodology renders the mapping sound and consistent.

The reason for which soundness and consistency are considered together is that they are closely related. In some cases (not all) soundness follows from consistency. The following corollary states a relationship between consistency and soundness of a mapping relation.

Corollary 1. *If ξ maps only pairs of categories c_i and c_j such that $c_i \nearrow c_j$ and ξ is consistent, than ξ is also sound.*

Proof. Let $c_i, c_j \in \mathrm{C}_1$ such that $c_i \nearrow_1 c_j$. By consistency, it must be that $\rho^{c_i \to c_j} \equiv \rho^{\xi(c_i) \to \xi(c_j)}$, that requires that $Dom(\rho^{c_i \to c_j}) = Dom(\rho^{\xi(c_i) \to \xi(c_j)})$ and $Codom(\rho^{c_i \to c_j}) = Codom(\rho^{\xi(c_i) \to \xi(c_j)})$. The conditions are equivalent to $\mathfrak{m}(c_i) = \mathfrak{m}(\xi(c_i))$ and $\mathfrak{m}(c_j) = \mathfrak{m}(\xi(c_j))$; thus soundness is proved. □

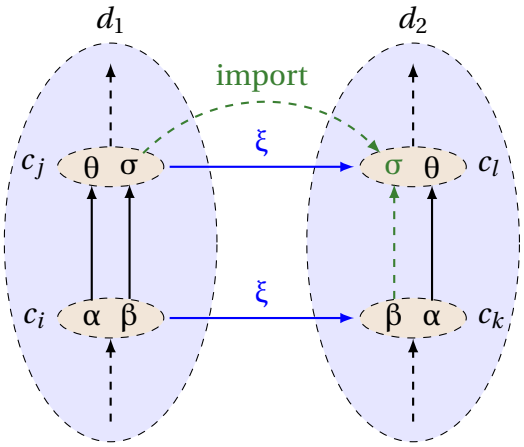


Figure 5.2: No sound → sound mapping

Corollary 2. *If* ξ *is a perfect mapping, then* $\xi^\sharp$ *is also a perfect mapping.*

Proof. The proof follows from Theorems 1 and 2. □

5.2 Checking Homogeneity

Unfortunately, homogeneity is not preserved when integrating different DW dimensions. Not even the case when integrating two homogeneous dimensions can ensure that the derived dimension is homogeneous. For example, in Figure 5.3 the mapping ξ establishes semantic equivalences among categories belonging to the homogeneous dimensions d_1 and d_2. When importing members from dimension d_1 to dimension d_2, the newly derived dimension, $d_2^\#$ is heterogeneous (the member γ has no equivalent member m_j in the category c_j'' such that $\gamma <_2 m_j$ and rolls-up directly to a member in category c_k'').

Interestingly, heterogeneity is also not preserved. There may be the case when a homogeneous dimension is obtained when integrating two heterogeneous dimensions. For example, in Figure 5.4 when integrating members from dimension d_1 to dimension d_2 (both heterogeneous), the newly derived dimension $d_2^\#$ is homogeneous. In this later case, the instance of dimension d_2 is completed with information from d_1. The same result may be obtained when integrating dimension members from dimension d_2 to dimension d_1, and the resulting dimension $d_1^\sharp$ would also be homogeneous.

A situation like the one described in Figure 5.4 may be encountered in real life cases when analysts decide to model the same information differently, or when information is partially missing by choice or by error. For example, categories c_j and c_j' may represent the region of a city (categories c_i and c_i'), that

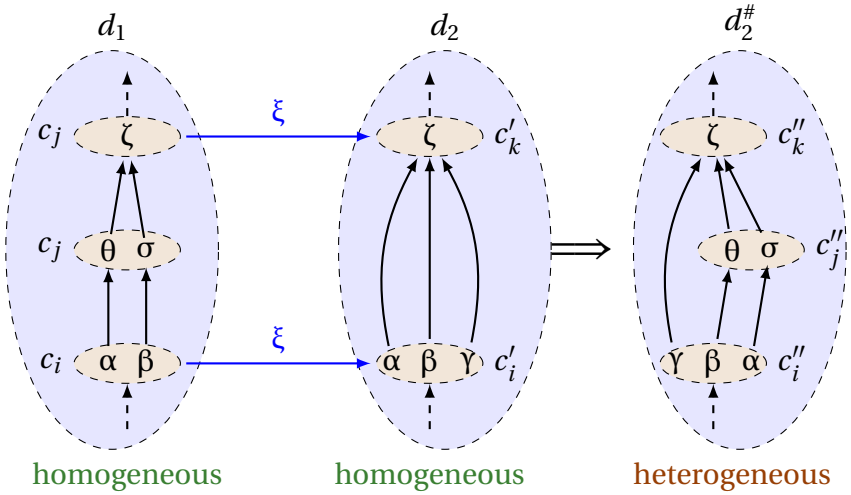


Figure 5.3: Homogeneous → Heterogeneous

was omitted for some cities in d_1 (member β) or for other members in d_2 (member α). The members of the categories are completed with instance data of the mapped categories.

Generally speaking, by integrating a combination of n homogeneous and heterogeneous dimensions, the resulting dimension may be either homogeneous or heterogeneous. Homogeneity thus is not preserved, but for some specific cases. The following theorem provides a sufficient condition for guaranteeing homogeneity.

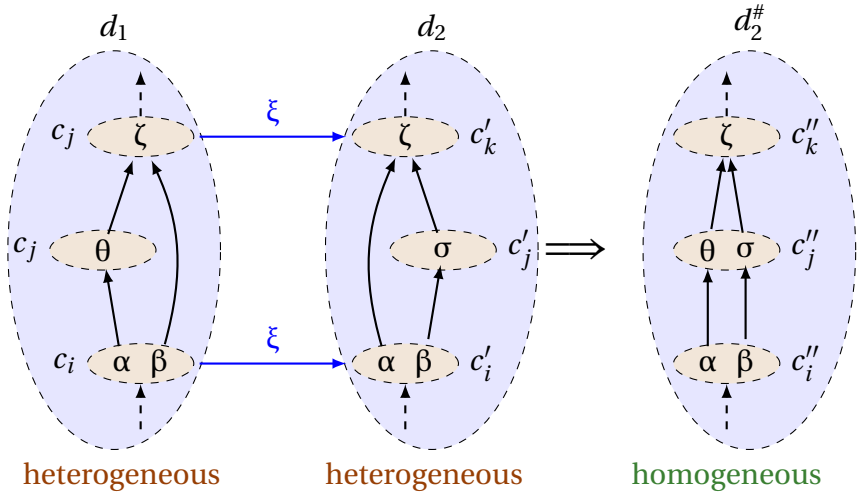


Figure 5.4: Heterogeneous → Homogeneous

Theorem 3. *If d_1 and d_2 are homogeneous and $\mathfrak{m}(c_k) \subseteq \mathfrak{m}(c_i)$, then $d_2^\natural$ is also homogeneous.*

Proof. Consider the importation rule represented in Figure 4.11. From $\mathfrak{m}(c_k) \subseteq \mathfrak{m}(c_i)$ it follows that for all $m_k \in \mathfrak{m}(c_k)$, there is a member $m_i \in \mathfrak{m}(c_i)$ such that $m_i = m_k$. Since d_1 is homogeneous, there is also $m_j \in \mathfrak{m}(c_j)$ such that $m_i <_1 m_j$. By construction, the category and member importation steps build a new dimension $d_2^\#$ such that $c_j \in \mathrm{C}_2$ (c_j in d_2 will be named c'_j, to avoid confusion), $c_k \nearrow_2^\# c'_j$ and $m_j \in \mathfrak{m}(c'_j)$ such that $m_k <_2^\# m_j$.

We have thus proved that in the dimension $d_2^\#$, for all $m_k \in \mathfrak{m}(c_k)$ there is a member $m_j \in \mathfrak{m}(c'_j)$ such that $m_k \nearrow_2^\# m_j$. Thus homogeneity is preserved. □

Corollary 3. *If d_1 and d_2 are in dimensional normal form and $\mathfrak{m}(c_k) \subseteq \mathfrak{m}(c_i)$, then $d_2^\#$ is also in dimensional normal form.*

Proof. Assuming to be working under the *closed-world* assumption and assuming only one *bottom* category per dimension, the proof follows from Theorem 3. □

An interesting observation may be drawn from Theorem 3. It turns out that when integrating two homogeneous dimensions d_1 and d_2 with bottom categories c_{bottom_1} and c_{bottom_2}, if $\mathfrak{m}(c_{\text{bottom}_1}) = \mathfrak{m}(c_{\text{bottom}_2})$, then by importing categories and members from one dimension to another, the newly obtained dimensions $d_1^\#$ and $d_2^\#$ are identical, apart from the names of the categories. In other words, there will be a total mapping $\kappa : \mathrm{C}_1^\# \to \mathrm{C}_2^\#$ that is *perfect*. Furthermore, κ^{-1} is also a *perfect* mapping. The newly derived dimensions $d_1^\#$ and $d_2^\#$ are identical to the one derived in [Torlone, 2008] by using the *tightly coupled* approach.

Corollary 4. *If $\mathfrak{m}(m_k) \nsubseteq \mathfrak{m}(c_i)$ and $c_j \notin \mathrm{C}_2$, then $d_2^\#$ is heterogeneous.*

Proof. If $\mathfrak{m}(m_k) \nsubseteq \mathfrak{m}(c_i)$, then there is $m_k \in \mathfrak{m}(c_k)$ such that $m_k \notin \mathfrak{m}(c_i)$. Implicitly, there is no $m_j \in \mathfrak{m}(c_j)$ such that $m_k <_1 m_j$. It follows by construction that $\nexists m_j \in \mathfrak{m}(c'_j)$ such that $m_j <_2^\# m_k$. Thus, $d_2^\#$ is heterogeneous. □

5.3 Slowly Changing Dimensions

One of the strongest assumptions in Data Warehousing is that dimensions are always static, that is the roll-up relations remain constant during the life-cycle of the DW. Although this is mostly true regarding the dimension hierarchy, it may happen that roll-up relations among dimension members change over

time, due mainly to the evolution of the application domain that the DW models. For example, a dimension hierarchy modeling persons and departments may remain constant during time (a person will always belong to a department), however it may happen that a person changes department over time. When such rare events occur, the dimension is said to be changing slowly (thus, the term *Slowly Changing Dimensions* was introduced by Ralph Kimball [Kimball, 1996]).

The change that occurs first in the operational database has to be propagated in the DW; the way the DW is updated changes the way analysts view current and historical data. Without analyzing all the possible scenarios from a technical point of view, the current section presents the semantics of the update policies when dealing with slowly changing dimensions.

In [Kimball and Ross, 2002] three main types of slowly changing dimensions are analyzed, Type 1, Type 2, Type 3.

Type 1 is the most simple and direct, and simply overwrites the old value with the new. If a person belongs to a department X, and at some point in time changes department to Y, then the change is propagated to the DW, where the person will belong to the department Y. All history regarding the person and department X will be lost, as the DW will look like the person will have always belonged to department Y. The approach sacrifices history in favor of simplicity, as all facts computed by person whilst belonging to department X will be now attributed to department Y.

Type 2 slowly changing dimension assumes inserting a new element in the DW that reflect the change in the operational database. The “new" person will have a different ID and from the analysis point of view will be completely distinct from the old one and all new facts will be correctly attributed to the new element belonging to department Y. The problem remains, however, when attempting to analyze all the person’s facts, as they will be divided among the old and the new instance of the person. For the purpose of the DW, the two instances will refer to two distinct persons.

Type 3 was conceived to surpass the limits of the Type 2 approach, and implies adding an extra reference that discriminates between actual and historical data. A person in this case will have an "actual department" and a "past department". This way it will be possible to correlate the facts related to that specific person, and still analyze them by correctly dividing by the department in use.

The dimension integration presented in the current thesis is independent

of the types of slowly changing dimensions presented earlier. That is, any of the three types may be adopted, and the changes will be propagated from individual operational database to the local DW, and further to the global DW. The only limitation is that the type of solution adopted must be coherent among overlapping instances of different DWs. In the previous person-department example, if the designer of a local DW has chosen a Type 1 approach for a specific person, then the same solution must be used for every instance of another DW containing data of the same person. If, for example, the same person is updated in tho distinct DWs by using a Type 1 and a Type 2 approach, then the information regarding that specific person would be aggregated incorrectly between the two DWs.

Chapter 6

Conclusions

The purpose of the research activity presented in this thesis was to tackle the Data Warehouse integration problem, a problem that is and will continue to be relevant in the dynamical economic context that sees many company merges and acquisitions, and the formation of new, innovative and dynamic business structures, like Business Intelligence Networks, virtual enterprises, federations of businesses or networks of co-opetition.

The Data Warehouse integration problem may be considered as a Data Integration context specific case, where analysts and developers can benefit from the decades of research in this research area. However, as was demonstrated in this work, context specific solutions must be applied in order to obtain more efficient results both in terms of mapping discovery as for the actual data integration step.

The current thesis provided two main contributions to the research area.

First of all, a complete Data Warehouse dimension integration methodology was presented. The methodology starts by analyzing dimensions, in particular the graph-like structure they induce on the dimension categories sets, and exploits recurrent knowledge of the structures to propose mappings among the dimension categories. An important observation to be made on the approach is that the mapping generation step depends on the completeness of the dimension instance which instead determine more accurate results. As Data Warehouse instances are not always complete, certain variability has been allowed in order to not compromise the mapping generation step.

The generated mappings are validated by using the Combined WordSense Disambiguation technique, a semantic approach that exploits the categories' labels and a lexical database (in this case, WordNet). The reason for using semantic validation was to increase accuracy by discarding improbable mappings. It has been proven in data integration that combined approaches generally provide better results than individual approaches.

The efficiency of the approach was proven with good results through experimental evaluation on several dimensions of real Data Warehouses. The results showed that even when the instances of the dimensions are completely disjoint, the approach is still able to correctly propose mapping between dimension categories.

The actual integration between Data Warehouse dimensions is done both at schema as at instance level. In some integration architectures, the integration of the dimension schema can provide not only the conceptual “union" of two dimensions, but can also increase the type and amount of information contained in each individual dimension. This means that apart from being able to query all the DWs contemporaneously, analysts now have the possibility of accessing the local Data Warehouse with increased querying capabilities.

The instance integration step can be executed through either of two approaches: by using the *d-chase* algorithm or by using an approximate approach, based on the *RELEVANT* clustering technique. The first is suggested in the rare cases where dimensions instances are created by using vocabularies that guarantee that identical values are used to represent the same concept of interest. If that is not the case, than the *RELEVANT* based approach is more appropriate. This later case creates sets of values (called clusters) based on similarity measures, like dominance and syntactic and semantic similarity. Members of categories that roll-up are forced to belong to the same clusters in order to maintain the similarity of the values.

One important result of the proposed approach, as presented in Section 4.6, is that the integration methodology is architecture independent. It can be thus used, depending on the project requirements, on several integration architectures that best fits the integration project specifications.

The second contribution of the thesis is the analysis of data quality properties that Data Warehouse dimensions have and maintain after the integration phase.

Homogeneity and heterogeneity is conceptually divided between inter-schema and intra-schema, and is analyzed before and after the integration phase. As was explained in Chapter 5, intra-schema homogeneity of the final dimension does not depend on the same property of the original dimensions it was obtained from. This means that by integrating two homogeneous dimensions, a heterogeneous dimension may be obtained. Strangely more, a homogeneous dimension may be obtained by integrating two heterogeneous dimensions. This is to state that homogeneity is neither a necessary nor a sufficient condition for obtaining a homogeneous dimension.

A specific condition regarding base categories and its members is, however, presented for *maintaining* the homogeneity of the dimensions after the schema and instance integration step. The reason for considering base cat-

egories is that the integration approach is done from a starting category upwards, through series of functional-dependency-like conditions. The same condition may be used from any other given starting category upwards. That is, if in some dimensions d_1 and d_2 the homogeneity condition is violated from some categories c_1 and c_2, then if the methodology generated a semantic mapping on some more detailed categories of c_1 and c_2 that contain the same members, then the resulting dimensions are also homogeneous. The condition is conceptually identical to the one regarding base categories.

One important study of the thesis regarder three dimension mapping properties previously presented in the research literature, namely coherency, soundness and consistency. While coherency states the correctness of the roll-up relations between pairs of mapped categories, soundness and consistency are properties describing the instance of the dimension (the members of the categories) and the partial order relations on the member set, that has to be maintained by the mapping. The violation of any of these properties is indcation that the mapping is incorrect, and integrating information using the mappings may lead to incorrect interpretations.

One important proof presented in the thesis is that the mapping generation step computes only coherent mappings, a result that confirms the accuracy and corectness of the discovered mappings. Furthermore, a theorem proves that soundness and consistency are maintained after the integration procedure.

As soundness and consistency are properties regarding the dimension instance, it is not surprising that they are correlated. In fact, as Corollary 1 demonstrates, in specific cases soundness follows from consistency.

Bibliography

[Aberer, 2011] Aberer, K. (2011). *Peer-to-Peer Data Management*, volume 3 of *Synthesis Lectures on Data Management*. Morgan & Claypool Publishers.

[Abiteboul et al., 1995] Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley.

[Acs and Yeung, 1999] Acs, Z. J. and Yeung, B. (1999). *Small and Medium-Sized Enterprises in the Global Economy*. University of Michigan Press.

[Aggarwal and Yu, 2008] Aggarwal, C. C. and Yu, P. S. (2008). Privacy-Preserving Data Mining: A Survey. In Gertz, M. and Jajodia, S., editors, *Handbook of Database Security*, pages 431–460. Springer US, Boston, MA.

[Akinde et al., 2002] Akinde, M. O., Böhlen, M. H., Johnson, T., Lakshmanan, L. V. S., and Srivastava, D. (2002). Efficient OLAP Query Processing in Distributed Data Warehouses. In *8th International Conference on Extending Database Technology on Advances in Database Technology - EDBT*, volume 2287 of *Lecture Notes in Computer Science*, pages 336–353, Berlin, Heidelberg. Springer-Verlag.

[Alqarni and Pardede, 2012] Alqarni, A. A. and Pardede, E. (2012). Integration of Data Warehouse and Unstructured Business Documents.

[Arenas et al., 2013] Arenas, M., Barceló, P., Fagin, R., and Libkin, L. (2013). Solutions and query rewriting in data exchange. *Information and Computation*, 228-229:28–61.

[Atkinson et al., 1992] Atkinson, M. P., Bancilhon, F., DeWitt, D. J., Dittrich, K. R., Maier, D., and Zdonik, S. B. (1992). The Object-Oriented Database System Manifesto. In *Building an Object-Oriented Database System, The Story of O2*, pages 3–20.

[Atwood, 1985] Atwood, T. (1985). An Object-Oriented DBMS for Design Support Applications. In *Proceedings of the IIEEE 1985 compint : computer aided technologies*, pages 299–307. IEEE Computer Society.

[Ballou and Tayi, 1999] Ballou, D. P. and Tayi, G. K. (1999). Enhancing data quality in data warehouse environments. *Communications of the ACM*, 42(1):73–78.

[Banek et al., 2006] Banek, M., Tjoa, A. M., and Stolba, N. (2006). Integrating Different Grain Levels in a Medical Data Warehouse Federation. In *8th International Conference on Data Warehousing and Knowledge Discovery - DaWaK (LNCS 4081)*, volume 4081, pages 185–194, Krakow, Poland. Springer Berlin Heidelberg.

[Banek et al., 2007] Banek, M., Vrdoljak, B., Tjoa, A. M., and Skocir, Z. (2007). Automating the Schema Matching Process for Heterogeneous Data Warehouses. In *9th International Conference on Data Warehousing and Knowledge Discovery - DaWaK 2007*, pages 45–54, Regensburg, Germany. Springer.

[Banek et al., 2008] Banek, M., Vrdoljak, B., Tjoa, A. M., and Skocir, Z. (2008). Automated Integration of Heterogeneous Data Warehouse Schemas. *International Journal of Data Warehousing and Mining (IJDWM)*, 4(4):1–21.

[Batini et al., 1986] Batini, C., Lenzerini, M., and Navathe, S. B. (1986). A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364.

[Beneventano et al., 2000] Beneventano, D., Bergamaschi, S., Castano, S., Corni, A., Guidetti, R., Malvezzi, G., Melchiori, M., and Vincini, M. (2000). Information Integration: The MOMIS Project Demonstration. In Abbadi, A. E., Brodie, M. L., Chakravarthy, S., Dayal, U., Kamel, N., Schlageter, G., and Whang, K.-Y., editors, *VLDB*, pages 611–614. Morgan Kaufmann.

[Beneventano et al., 2003] Beneventano, D., Bergamaschi, S., Guerra, F., and Vincini, M. (2003). Synthesizing an Integrated Ontology. *IEEE Internet Computing*, 7(5):42–51.

[Beneventano et al., 2013] Beneventano, D., Olaru, M.-O., and Vincini, M. (2013). Analyzing Dimension Mappings and Properties in Data Warehouse Integration. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences - Confederated International Conferences: CoopIS, DOA-Trusted Cloud, and ODBASE 2013 (LNCS 8185)*, volume 8185 of *Lecture Notes in Computer Science*, pages 616–623, Graz, Austria. Springer Berlin Heidelberg.

[Bergamaschi et al., 2011] Bergamaschi, S., Beneventano, D., Guerra, F., and Orsini, M. (2011). Data Integration. In Embley, D. and Thalheim, B., editors, *Handbook of Conceptual Modeling*, chapter 14, pages 441–476.

[Bergamaschi et al., 2007a] Bergamaschi, S., Bouquet, P., Giacomuzzi, D., Guerra, F., Po, L., and Vincini, M. (2007a). An Incremental Method for the Lexical Annotation of Domain Ontologies. *Int. J. Semantic Web Inf. Syst.*, 3(3):57–80.

[Bergamaschi et al., 1999] Bergamaschi, S., Castano, S., and Vincini, M. (1999). Semantic Integration of Semistructured and Structured Data Sources. *SIGMOD Record*, 28(1):54–59.

[Bergamaschi et al., 2001] Bergamaschi, S., Castano, S., Vincini, M., and Beneventano, D. (2001). Retrieving and integrating data from multiple sources: the MOMIS approach. *Data Knowl. Eng.*, 36(3):215–249.

[Bergamaschi et al., 2010] Bergamaschi, S., Guerra, F., Orsini, M., Sartori, C., and Vincini, M. (2010). A Semantic Approach to ETL Technologies. *Data & Knowledge Engineering*, 70(8):717–731.

[Bergamaschi et al., 2012] Bergamaschi, S., Olaru, M.-O., Sorrentino, S., and Vincini, M. (2012). Dimension matching in Peer-to-Peer Data Warehousing. In *16th IFIP WG 8.3 International Conference on Decision Support Systems*, pages 149–160, Anávissos, Greece.

[Bergamaschi et al., 2007b] Bergamaschi, S., Orsini, M., Guerra, F., and Sartori, C. (2007b). Relevant values: New metadata to provide insight on attribute values at schema level. In *Proceedings of the Ninth International Conference on Enterprise Information Systems - ICEIS 2007*, pages 274–279, Funchal, Portugal.

[Bergamaschi et al., 2007c] Bergamaschi, S., Po, L., Sala, A., and Sorrentino, S. (2007c). Data source annotation in data integration systems. In *Proceedings of The Fifth International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, Vienna, Austria.

[Bergamaschi et al., 2007d] Bergamaschi, S., Po, L., and Sorrentino, S. (2007d). Automatic Annotation in Data Integration Systems. In *OTM Workshops (1)*, pages 27–28.

[Bergamaschi et al., 2008] Bergamaschi, S., Po, L., and Sorrentino, S. (2008). Automatic annotation for mapping discovery in data integration systems. In *Proceedings of the Sixteenth Italian Symposium on Advanced Database Systems - SEBD*, pages 334–341, Mondello, Italy.

[Bergamaschi et al., 2007e] Bergamaschi, S., Sartori, C., Guerra, F., and Orsini, M. (2007e). Extracting Relevant Attribute Values for Improved Search. *IEEE Internet Computing*, 11(5):26–35.

[Berger and Schrefl, 2006] Berger, S. and Schrefl, M. (2006). Analysing multidimensional data across autonomous data warehouses. In *Data Warehousing and Knowledge Discovery*, pages 120–133.

[Berger and Schrefl, 2008] Berger, S. and Schrefl, M. (2008). From Federated Databases to a Federated Data Warehouse System. In *Proceedings of the 41st Hawaii International International Conference on Systems Science, HICS 2008*, page 394.

[Bernstein et al., 2002] Bernstein, P. A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., and Zaihrayeu, I. (2002). Data Management for Peer-to-Peer Computing : A Vision. In *Fifth International Workshop on the Web and Databases, WebDB 2002*, pages 89–94.

[Blanco et al., 1994] Blanco, J. M., Illaramendi, A., and Alfredo, G. (1994). Building a Federated Relational Database System: An Approach Using a Knowledge-Based System. *International Journal of Cooperative Information Systems*, 03(04):415–455.

[Blili and Raymond, 1993] Blili, S. and Raymond, L. (1993). Information technology: Threats and opportunities for small and medium-sized enterprises. *International Journal of Information Management*, 13(6):439–448.

[Boussaid et al., 2006] Boussaid, O., Messaoud, R. B., Choquet, R., and Anthoard, S. (2006). X-Warehousing: An XML-Based Approach for Warehousing Complex Data. In *Advances in Databases and Information Systems, Lecture Notes in Computer Science*, volume 4152, pages 39–54, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Brandenburger and Nalebuff, 1996] Brandenburger, A. and Nalebuff, B. (1996). *Co-opetition*. New York, NY, USA.

[Bray and DeRose, 1997] Bray, T. and DeRose, S. J. (1997). Extensible Markup Language (XML) Part 2: Linking. *World Wide Web Journal*, 2(4):67–82.

[Bray et al., 1997] Bray, T., Paoli, J., and Sperberg-McQueen, C. M. (1997). Extensible Markup Language (XML). *World Wide Web Journal*, 2(4):27–66.

[Cabibbo and Torlone, 1998] Cabibbo, L. and Torlone, R. (1998). A Logical Approach to Multidimensional Databases. In Schek, H.-J., Alonso, G., Saltor,

F., and Ramos, I., editors, *Advances in Database Technology - EDBT'98, 6th International Conference on Extending Database Technology*, volume 1377 of *Lecture Notes in Computer Science*, pages 183–197, Valencia, Spain. Springer Berlin Heidelberg.

[Cabibbo and Torlone, 2004a] Cabibbo, L. and Torlone, R. (2004a). Dimension Compatibility for Data Mart Integration. In Agosti, M., Dessì, N., and Schreiber, F., editors, *Proceedings of the Twelfth Italian Symposium on Advanced Database Systems - SEBD 2004*, pages 6–17, S. Margherita di Pula, Cagliari, Italy.

[Cabibbo and Torlone, 2004b] Cabibbo, L. and Torlone, R. (2004b). On the Integration of Autonomous Data Marts. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004)*, pages 223–231, Santorini Island, Greece. IEEE Computer Society.

[Cabibbo and Torlone, 2005] Cabibbo, L. and Torlone, R. (2005). Integrating heterogeneous multidimensional databases. In *Proceedings of the17th International Conference on Scientific and Statistical Database Management, SSDBM 2005*, pages 205–214, Santa Barbara, CA, USA.

[Calì et al., 2003] Calì, A., Lembo, D., Lenzerini, M., and Rosati, R. (2003). Source Integration for Data Warehousing. In *Multidimensional Databases*, pages 361–392.

[Calvanese et al., 2001] Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., and Rosati, R. (2001). Data Integration in Data Warehousing. *International Journal of Cooperative Information Systems*, 10(3):237–271.

[Calvanese et al., 1999] Calvanese, D., Giacomo, G. D., Lenzerini, M., Nardi, D., and Rosati, R. (1999). A Principled Approach to Data Integration and Reconciliation in Data Warehousing. In *DMDW*, page 16.

[Camarinha-Matos et al., 2009] Camarinha-Matos, L. M., Afsarmanesh, H., Galeano, N., and Molina, A. (2009). Collaborative networked organizations – Concepts and practice in manufacturing enterprises. *Computers & Industrial Engineering*, 57(1):46–60.

[Carey et al., 1995] Carey, M. J., Haas, L. M., Schwarz, P. M., Arya, M., Cody, W. F., Fagin, R., Flickner, M., Luniewski, A., Niblack, W., Petkovic, D., II, J. T., Williams, J. H., and Wimmers, E. L. (1995). Towards Heterogeneous Multimedia Information Systems: The Garlic Approach. In *ifth International Workshop on Research Issues in Data Engineering - Distributed Object Management - RIDE-DOM*, pages 124–131, Taipei, Taiwan. IEEE Computer Society.

[Casler, 1992] Casler, S. D. (1992). *Introduction to Economics*. Collins Reference, New York, NY, USA.

[Ceri and Widom, 1993] Ceri, S. and Widom, J. (1993). Managing Semantic Heterogeneity with Production Rules and Persistent Queues. In *19th International Conference on Very Large Data Bases - VLDB 1993*, pages 108–119, Dublin, Ireland. Morgan Kaufmann.

[Chawathe et al., 1994] Chawathe, S. S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. D., and Widom, J. (1994). The TSIMMIS Project: Integration of Heterogeneous Information Sources. In *10th Meeting of the Information Processing Society of Japan*, pages 7–18.

[Chen, 1976] Chen, P. P.-S. (1976). The entity-relationship model - toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36.

[Codd, 1970] Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

[Czejdo et al., 1987] Czejdo, B. D., Rusinkiewicz, M., and Embley, D. W. (1987). An Approach to Schema Integration and Query Formulation in Federated Database Systems. In *Proceedings of the Third International Conference on Data Engineering - ICDE 1987*, pages 477–484, Los Angeles, California. IEEE Computer Society.

[Devlin, 1996] Devlin, B. (1996). *Data Warehouse: From Architecture to Implementation*. Addison-Wesley Professional.

[Dhamankar et al., 2004] Dhamankar, R., Lee, Y., Doan, A., Halevy, A., and Domingos, P. (2004). iMAP: Discovering Complex Mappings between Database Schemas. In Weikum, G., König, A. C., and Deß loch, S., editors, *Proceedings of the 2004 ACM SIGMOD international conference on Management of data - SIGMOD '04*, pages 383–394, Paris, France. ACM Press.

[Doan et al., 2000] Doan, A., Domingos, P., and Levy, A. Y. (2000). Learning Source Description for Data Integration. In *Third International Workshop on the Web and Databases - WebDB 2000*, pages 81–86.

[Doan and Halevy, 2005] Doan, A. and Halevy, A. Y. (2005). Semantic Integration Research in the Database Community: A Brief Survey. *AI Magazine*, 26(1):83–94.

[Euzenat and Shvaiko, 2007] Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Fagin et al., 2005a] Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. (2005a). Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124.

[Fagin et al., 2005b] Fagin, R., Kolaitis, P. G., and Popa, L. (2005b). Data exchange: getting to the core. *ACM Transactions on Database Systems*, 30(1):174–210.

[Frakes and Baeza-Yates, 1992] Frakes, W. B. and Baeza-Yates, R. (1992). *Information Retrieval: Data Structures and Algorithms*. Prentice Hall.

[Friedman et al., 1999] Friedman, M., Levy, A. Y., and Millstein, T. D. (1999). Navigational Plans for Data Integration. In *Intelligent Information Integration*.

[Galhardas et al., 2001] Galhardas, H., Florescu, D., Shasha, D., Simon, E., and Saita, C.-A. (2001). Declarative Data Cleaning: Language, Model, and Algorithms. In *27th International Conference on Very Large Data Bases - VLDB*, pages 371–380, Rome, Italy. Morgan Kaufmann.

[Golfarelli et al., 1998a] Golfarelli, M., Maio, D., and Rizzi, S. (1998a). Conceptual design of data warehouses from E/R schemes. In *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, volume 7, pages 334–343. IEEE Comput. Soc.

[Golfarelli et al., 1998b] Golfarelli, M., Maio, D., and Rizzi, S. (1998b). The Dimensional Fact Model: A Conceptual Model for Data Warehouses. *Int. J. Cooperative Inf. Syst.*, 7(2-3):215–247.

[Golfarelli et al., 2010] Golfarelli, M., Mandreoli, F., Penzo, W., Rizzi, S., and Turricchia, E. (2010). Towards OLAP query reformulation in Peer-to-Peer Data Warehousing. In Song, I.-Y. and Ordonez, C., editors, *DOLAP*, pages 37–44. ACM.

[Golfarelli et al., 2011] Golfarelli, M., Mandreoli, F., Penzo, W., Rizzi, S., and Turricchia, E. (2011). OLAP Query Reformulation in Peer-to-Peer Data Warehousing. *Information Systems*, 37(5):393–411.

[Golfarelli and Rizzi, 1998] Golfarelli, M. and Rizzi, S. (1998). A methodological framework for data warehouse design. In *Proceedings of the 1st ACM international workshop on Data warehousing and OLAP - DOLAP '98*, pages 3–9, New York, New York, USA. ACM Press.

[Golfarelli and Rizzi, 2006] Golfarelli, M. and Rizzi, S. (2006). *Data Warehouse: teoria e prattica della progettazione*. McGraw-Hill, Milano, second edition.

[Golfarelli et al., 2001] Golfarelli, M., Rizzi, S., and Vrdoljak, B. (2001). Data warehouse design from XML sources. In *Proceedings of the 4th ACM international workshop on Data warehousing and OLAP - DOLAP '01*, pages 40–47, New York, New York, USA. ACM Press.

[Gray et al., 1997] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H. (1997). Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-Tab, and Sub Totals. *Data Min. Knowl. Discov.*, 1(1):29–53.

[Guerra et al., 2012] Guerra, F., Olaru, M.-O., and Vincini, M. (2012). Mapping and Integration of Dimensional Attributes Using Clustering Techniques. In *13th International Conference on E-Commerce and Web Technologies*, volume 123 of *Lecture Notes in Business Information Processing*, pages 38–49, Vienna, Austria. Springer.

[Halevy, 2001] Halevy, A. Y. (2001). Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294.

[Halevy et al., 2003] Halevy, A. Y., Ives, Z. G., Suciu, D., and Tatarinov, I. (2003). Schema Mediation in Peer Data Management Systems. In IEEE Computer Society, editor, *9th International Conference on Data Engineering - ICDE 2003*, pages 505–516, Bangalore, India.

[Hammer and McLeod, 1993] Hammer, J. and McLeod, D. (1993). An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems. *International Journal of Cooperative Information Systems*, 02(01):51–83.

[Harinarayan et al., 1996] Harinarayan, V., Rajaraman, A., and Ullman, J. D. (1996). Implementing data cubes efficiently. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data - SIGMOD '96*, pages 205–216, New York, New York, USA. ACM Press.

[Hsiao, 1992a] Hsiao, D. K. (1992a). Federated Databases and Systems: Part I - A Tutorial on Their Data Sharing. *VLDB J.*, 1(1):127–179.

[Hsiao, 1992b] Hsiao, D. K. (1992b). Federated Databases and Systems: Part II - A Tutorial on Their Resource Consolidation. *VLDB J.*, 1(2):285–310.

[Hsiao et al., 1990] Hsiao, D. K., Kamel, M. N., and Wu, C. T. (1990). The Federated Databases and System: A New Generation of Advanced Database Systems. In *The Federated Databases and System: A New Generation of Advanced Database Systems - DEXA*, pages 186–190.

[Hull, 1997] Hull, R. (1997). Managing semantic heterogeneity in databases. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems - PODS '97*, pages 51–61, New York, New York, USA. ACM Press.

[Hurtado et al., 2005] Hurtado, C. A., Gutierrez, C., and Mendelzon, A. O. (2005). Capturing summarizability with integrity constraints in OLAP. *ACM Transactions on Database Systems*, 30(3):854–886.

[Hurtado and Mendelzon, 2001] Hurtado, C. A. and Mendelzon, A. O. (2001). Reasoning about Summarizability in Heterogeneous Multidimensional Schemas. In Bussche, J. and Vianu, V., editors, *International Conference on Database Theory - ICDT*, volume 1973 of *Lecture Notes in Computer Science*, pages 375–389. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Hurtado and Mendelzon, 2002] Hurtado, C. A. and Mendelzon, A. O. (2002). OLAP dimension constraints. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '02*, page 169, New York, New York, USA. ACM Press.

[Inmon, 2002] Inmon, W. H. (2002). *Building the data warehouse*. John Wiley & Sons, Inc., third edition.

[Jagadish et al., 1999] Jagadish, H. V., Lakshmanan, L. V. S., and Srivastava, D. (1999). What can Hierarchies do for Data Warehouses? In *25th International Conference on Very Large Data Bases, September - VLDB*, pages 530–541, Edinburgh, Scotland. Morgan Kaufmann.

[Jarke et al., 2002] Jarke, M., Lenzerini, M., Vassiliou, Y., and Vassiliadis, P. (2002). *Fundamentals of Data Warehouses*. Springer.

[Kamel and Kamel, 1992] Kamel, M. N. and Kamel, N. N. (1992). Federated database management system: Requirements, issues and solutions. *Computer Communications*, 15(4):270–278.

[Kedad and Métais, 1999] Kedad, Z. and Métais, E. (1999). Dealing with Semantic Heterogeneity During Data Integration. In Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., and Métais, E., editors, *Conceptual Modeling - ER '99, 18th International Conference on Conceptual Modeling*, volume 1728 of

Lecture Notes in Computer Science, pages 325–339, Paris. Springer Berlin Heidelberg.

[Kern et al., 2011] Kern, R., Ryk, K., and Nguyen, N. T. (2011). A Framework for Building Logical Schema and Query Decomposition in Data Warehouse Federations. In *Third International Conference on Computational Collective Intelligence. Technologies and Applications - ICCCI 2011*, volume 6922 of *Lecture Notes in Computer Science*, pages 612–622, Gdynia, Poland. Springer Berlin Heidelberg.

[Kern et al., 2013] Kern, R., Stolarczyk, T., and Nguyen, N. T. (2013). A formal framework for query decomposition and knowledge integration in data warehouse federations. *Expert Systems with Applications*, 40(7):2592–2606.

[Khouri and Ladjel, 2010] Khouri, S. and Ladjel, B. (2010). A methodology and tool for conceptual designing a data warehouse from ontology-based sources. In *Proceedings of the ACM 13th international workshop on Data warehousing and OLAP - DOLAP '10*, page 19, New York, New York, USA. ACM Press.

[Kim et al., 1995] Kim, W., Choi, I., Gala, S. K., and Scheevel, M. (1995). On Resolving Schematic Heterogeneity in Multidatabase Systems. In *Modern Database Systems*, pages 521–550.

[Kim and Seo, 1991] Kim, W. and Seo, J. (1991). Classifying Schematic and Data Heterogeneity in Multidatabase Systems. *IEEE Computer*, 24(12):12–18.

[Kimball, 1996] Kimball, R. (1996). *The Data Warehouse Toolkit - Practical Techniques for Building Dimensional Data Warehouses*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition.

[Kimball and Ross, 2002] Kimball, R. and Ross, M. (2002). *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*, volume 32. John Wiley & Sons, Inc., New York, NY, USA.

[Lechtenbörger, 2001] Lechtenbörger, J. (2001). *Data Warehouse Schema Design*. Akademische Verlagsgesellschaft Aka Gmbh.

[Lehner et al., 1998] Lehner, W., Albrecht, J., and Wedekind, H. (1998). Normal Forms for Multidimensional Databases. In *10th International Conference on Scientific and Statistical Database Management - SSDBM*, pages 63–72, Capri, Italy. IEEE Computer Society.

[Lenz and Shoshani, 1997] Lenz, H.-J. and Shoshani, A. (1997). Summarizability in OLAP and Statistical Data Bases. In *Ninth International Conference on Scientific and Statistical Database Management - SSDBM*, pages 132–143, Olympia, Washington, USA. IEEE Computer Society.

[Lenzerini, 2002] Lenzerini, M. (2002). Data Integration: A Theoretical Perspective. In Popa, L., editor, *Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems - PODS*, pages 233–246, Madison, Wisconsin, USA. ACM.

[Levy et al., 2001] Levy, M., Löbbecke, C., and Powell, P. (2001). SMEs Coopetition and Knowledge Sharing: The IS Role. In *9th European Conference on Information Systems, Global Co-operation in the New Millennium - ECIS*, pages 640–652, Bled, Slovenia.

[Li and McLeod, 1991] Li, Q. and McLeod, D. (1991). An Object-Oriented Approach to Federated Databases. In *First International Workshop on Research Issues on Data Engineering: Interoperability in Multidatabase Systems RIDE-IMS*, pages 64–70, Kyoto, Japan. IEEE Computer Society.

[Loebecke et al., 1999] Loebecke, C., Van Fenema, P. C., and Powell, P. (1999). Co-opetition and knowledge transfer. *ACM SIGMIS Database*, 30(2):14–25.

[Lovász and Plummer, 1986] Lovász, L. and Plummer, M. D. (1986). *Matching Theory*. North Holland, Amsterdam.

[Madhavan et al., 2001] Madhavan, J., Bernstein, P. A., and Rahm, E. (2001). Generic Schema Matching with Cupid. In Apers, P. M. G., Atzeni, P., Ceri, S., Paraboschi, S., Ramamohanarao, K., and Snodgrass, R. T., editors, *VLDB*, pages 49–58. Morgan Kaufmann.

[Maleszka et al., 2012] Maleszka, M., Mianowska, B., and Nguyen, N. T. (2012). A framework for data warehouse federations building. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2897–2902, Seoul, Korea. IEEE Computer Society.

[Maurino et al., 2013] Maurino, A., Venturini, C., and Viscusi, G. (2013). Coopetitive Data Warehouse: A Case Study. In *Proceedings of the 25th International Conference on Advanced Information Systems Engineering - CAiSE 2013*, volume 7908 of *Lecture Notes in Computer Science*, pages 482–497, Berlin, Heidelberg. Springer Berlin Heidelberg.

[McFadden et al., 1998] McFadden, F. R., Hoffer, J. A., and Prescott, M. B. (1998). *Modern Database Management*. Addison-Wesley.

[McLeod and Heimbigner, 1980] McLeod, D. and Heimbigner, D. (1980). A federated architecture for database systems. In *Proceedings of the May 19-22, 1980, national computer conference on - AFIPS '80*, pages 283–289, Anaheim, USA. ACM Press.

[Melnik et al., 2002] Melnik, S., Garcia-Molina, H., and Rahm, E. (2002). Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *18th International Conference on Data Engineering - ICDE*, pages 117–128, San Jose, California, USA. IEEE Computer Society.

[Miller et al., 1994] Miller, R., Ioannidis, Y., and Ramakrishnan, R. (1994). Schema equivalence in heterogeneous systems: Bridging theory and practice. *Information Systems*, 19(1):3–31.

[Miller et al., 2001] Miller, R. J., Hernández, M. A., Haas, L. M., Yan, L.-L., Ho, C. T. H., Fagin, R., and Popa, L. (2001). The Clio Project: Managing Heterogeneity. *SIGMOD Record*, 30(1):78–83.

[Milo and Zohar, 1998] Milo, T. and Zohar, S. (1998). Using Schema Matching to Simplify Heterogeneous Data Translation. In *Proceedings of 24rd International Conference on Very Large Data Bases - VLDB*, pages 122–133, New York City, USA. Morgan Kaufmann.

[Mouhni and Elkalay, 2013] Mouhni, N. and Elkalay, A. (2013). Ontology based data warehouses federation management system. *Computer Research Repository - CoRR*, abs/1310.5.

[Mowshowitz, 1986] Mowshowitz, A. (1986). Social Dimensions of Office Automation. *Advances in Computers*, 25:335–404.

[Mowshowitz, 1997] Mowshowitz, A. (1997). Virtual organization. *Communications of the ACM*, 40(9):30–37.

[Mowshowitz, 2003] Mowshowitz, A. (2003). Virtual organization: toward a theory of societal transformation stimulated by information technology. *Ubiquity*, 2003(May):2.

[Navigli, 2009] Navigli, R. (2009). Word sense disambiguation. *ACM Computing Surveys*, 41(2):1–69.

[Neumayr et al., 2009] Neumayr, B., Grün, K., and Schrefl, M. (2009). Multi-Level Domain Modeling with M-Objects and M-Relationships. In *Sixth Asia-Pacific Conference on Conceptual Modelling*, pages 107–116, Wellington, New Zealand.

[Neumayr et al., 2010] Neumayr, B., Schrefl, M., and Thalheim, B. (2010). Hetero-homogeneous hierarchies in data warehouses. In *Proceedings of the Seventh Asia-Pacific Conference on Conceptual Modelling (APCCM 2010)*, volume 110.

[Object Managenemt Group, 2003] Object Managenemt Group (2003). *Common Warehouse MetaModel*. 1.1 edition.

[Olaru, 2012] Olaru, M.-O. (2012). Partial Multi-dimensional Schema Merging in Heterogeneous Data Warehouses. In Atzeni, P., Cheung, D., and Ram, S., editors, *31st International Conference on Conceptual Modeling - ER2012*, volume 7532 of *Lecture Notes in Computer Science*, pages 563–571, Florence, Italy. Springer Berlin Heidelberg.

[Olaru and Vincini, 2012] Olaru, M.-O. and Vincini, M. (2012). A Dimension Integration Method for a Heterogeneous Data Warehouse Environment. In *Proceedings of the International Conference on Data Communication Networking, e-Business and Optical Communication Systems - ICETE 2012*, pages 278–283, Rome, Italy.

[Palopoli et al., 1998] Palopoli, L., Saccà, D., and Ursino, D. (1998). Semi-Automatic Semantic Discovery of Properties from Database Schemas. In *International Database Engineering and Applications Symposium - IDEAS*, pages 244–253, Cardiff, Wales. IEEE Computer Society.

[Rafanelli and Shoshani, 1990] Rafanelli, M. and Shoshani, A. (1990). STORM: A Statistical Object Representation Model. In Michalewicz, Z., editor, *Statistical and Scientific Database Management - SSDBM*, volume 420 of *Lecture Notes in Computer Science*, pages 14–29, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Rahm and Bernstein, 2001] Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350.

[Rizzi et al., 2006] Rizzi, S., Abelló, A., Lechtenbörger, J., and Trujillo, J. (2006). Research in data warehouse modeling and design: dead or alive? In *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP - DOLAP '06*, page 3, New York, New York, USA. ACM Press.

[Rodríguez and Egenhofer, 2003] Rodríguez, M. A. and Egenhofer, M. J. (2003). Determining Semantic Similarity among Entity Classes from Different Ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):442–456.

[Romero and Abelló, 2009] Romero, O. and Abelló, A. (2009). A Survey of Multidimensional Modeling Methodologies. *International Journal of Data Warehousing and Mining*, 5(2):1–23.

[Roth et al., 1996] Roth, M. T., Arya, M., Haas, L., Carey, M., Cody, W., Fagin, R., Schwarz, P., Thomas, J., and Wimmers, E. (1996). The Garlic project. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data - SIGMOD '96*, page 557, New York, New York, USA. ACM Press.

[Schütz and Schrefl, 2011] Schütz, C. and Schrefl, M. (2011). Incremental integration of data warehouses: the hetero-homogeneous approach. In *DOLAP 2011, ACM 14th International Workshop on Data Warehousing and OLAP*.

[Sedlak and Tumbas, 2006] Sedlak, O. and Tumbas, P. (2006). Managing Knowledge by the Information System and Game-Theoretic Approach. *International Scientific Journal of Management Information Systems*, 1(1).

[Sheth, 1991] Sheth, A. P. (1991). Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. In *17th International Conference on Very Large Data Bases - VLDB 1991*, page 489, Barcelona, Spain. Morgan Kaufmann.

[Sheth and Larson, 1990] Sheth, A. P. and Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236.

[Shvaiko and Euzenat, 2005] Shvaiko, P. and Euzenat, J. (2005). *Journal on Data Semantics IV*, volume 3730 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Sorrentino, 2011] Sorrentino, S. (2011). *Label Normalization and Lexical Annotation for Schema and Ontology Matching*. PhD thesis, University of Modena and Reggio Emilia, Italy.

[Sorrentino et al., 2010] Sorrentino, S., Bergamaschi, S., Gawinecki, M., and Po, L. (2010). Schema label normalization for improving schema matching. *Data & Knowledge Engineering*, 69(12):1254–1273.

[Torlone, 2008] Torlone, R. (2008). Two approaches to the integration of heterogeneous data warehouses. *Distributed and Parallel Databases*, 23(1):69–97.

[Torlone, 2009] Torlone, R. (2009). Interoperability in Data Warehouses. In Liu, L. and Özsu, T., editors, *Encyclopedia of Database Systems*, pages 1560–1564. Springer, Berlin.

[Tria et al., 2013] Tria, F. D., Lefons, E., and Tangorra, F. (2013). Ontological Approach to Data Warehouse Source Integration. In *Information Sciences and Systems 2013 (LNEE 264)*, volume 264 of *Lecture Notes in Electrical Engineering*, chapter 25, pages 251–259. Springer International Publishing, Cham.

[Ullman, 1997] Ullman, J. D. (1997). Information Integration Using Logical Views. In Afrati, F. and Kolaitis, P., editors, *6th International Conference on Database Theory - ICDT '97*, volume 1186 of *Lecture Notes in Computer Science*, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Vassiliadis, 2009] Vassiliadis, P. (2009). A Survey of Extract-Transform-Load Technology. *International Journal of Data Warehousing and Mining (IJDWM)*, 5(3):1–27.

[Westlake, 1988] Westlake, A. (1988). Geographical Database Systems and Statistical Information -Panel. In *4th International Working Conference Working Conference on Statistical and Scientific Database Management*, volume 339 of *Lecture Notes in Computer Science*, pages 420–426, Rome, Italy. Springer-Verlag.

[Wiederhold, 1992] Wiederhold, G. (1992). Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49.

[Wu and Hå kansson, 2010] Wu, D. and Hå kansson, A. (2010). Applying a Knowledge Based System for Metadata Integration for Data Warehouses. In *Knowledge-Based and Intelligent Information and Engineering Systems - LNCS*, volume 6279, pages 60–69, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Wyss and Van Gucht, 2001] Wyss, C. and Van Gucht, D. (2001). A relational algebra for data/metadata integration in a federated database system. In *Proceedings of the tenth international conference on Information and knowledge management - CIKM'01*, pages 65–72, Atlanta, Georgia, USA. ACM Press.

[Zhu and Buchmann, 2002] Zhu, Y. and Buchmann, A. P. (2002). Evaluating and Selecting Web Sources as External Information Resources of a Data Warehouse. In *3rd International Conference on Web Information Systems Engineering - WISE*, pages 149–160, Singapore. IEEE Computer Society.