Distrib Parallel Databases (2008) 23: 69–97
DOI 10.1007/s10619-007-7022-z

Two approaches to the integration of heterogeneous data warehouses

Riccardo Torlone

Published online: 23 December 2007

Abstract In this paper we address the problem of integrating independent and possibly heterogeneous data warehouses, a problem that has received little attention so far, but that arises very often in practice.

We start by tackling the basic issue of matching heterogeneous dimensions and provide a number of general properties that a dimension matching should fulfill. We then propose two different approaches to the problem of integration that try to enforce matchings satisfying these properties. The first approach refers to a scenario of loosely coupled integration, in which we just need to identify the common information between data sources and perform join operations over the original sources. The goal of the second approach is the derivation of a materialized view built by merging the sources, and refers to a scenario of tightly coupled integration in which queries are performed against the view.

We also illustrate architecture and functionality of a practical system that we have developed to demonstrate the effectiveness of our integration strategies.

Keywords Data integration · Data warehouse · Data mart · Multidimensional database · Heterogeneous information

A preliminary version this paper appeared, under the title "Integrating Heterogeneous Multidimensional Databases" [9], in *17th Int. Conference on Scientific and Statistical Database Management, 2005*.

R. Torlone (✉)
Dipartimento di Informatica e Automazione, Università Roma Tre, Roma, Italy
e-mail: torlone@dia.uniroma3.it

1 Introduction

1.1 Motivations

Data warehousing can be considered today a mature technology. Many advanced tools exist to support business analysts in the rapid construction, the effective maintenance and the efficient analysis of data marts, the building blocks of a data warehouse. As it often happens, however, the rapid and uncontrolled spreading of these tools within organizations has led in many cases a number of previously uncharted and rather involved challenges.

One problem that needs to be solved in many practical cases is the integration of data marts that have been developed and operated independently. Indeed, a common practice for building a data warehouse is to implement a series of data marts, each of which provides a dimensional view of a single business process [17]. These data marts should be based on common dimensions and facts but, very often, different departments of the same company develop their data marts independently, and it turns out that their integration is a difficult task.

Actually, the need for combining autonomous data marts arises in other common scenarios. For instance, when different companies merge or get involved in a federated project or when there is the need to combine a proprietary data warehouse with data available elsewhere, for instance, in external (and likely heterogeneous) information sources or in multidimensional data wrapped from the Web.

1.2 Goal and contributions

The goal of this paper is the investigation of theoretical and practical issues related to the design of an integration tool for data warehouses, similar in spirit to other tools supporting the integration of heterogeneous data sources [23], but specific for multidimensional databases. In fact, we believe that this problem can be tackled in a systematic way, for two main reasons. First, multidimensional databases are structured in a rather uniform way, along the widely accepted notions of dimension and fact. Second, data quality in data warehouses is usually higher than in generic databases, since they are obtained by reconciling several data sources.

We start by addressing the basic problem of integrating a pair of autonomous dimensions. To this end, we have introduced the basic notion of *dimension compatibility*, which extends an earlier notion proposed by Kimball [17]. Intuitively, two dimensions of different data marts are compatible if their common information is consistent. In a preliminary study [8], we have shown that dimension compatibility gives the ability to correlate, in a correct way, multiple data marts by means of *drill-across queries* [17], which are basically joins, over common dimensions, of different data marts.

Building on this notion, we identify a number of desirable properties that a *matching* between dimensions (that is, a correspondence between their levels) should satisfy: (i) the *coherence* of the hierarchies on levels, (ii) the *soundness* of the paired levels, according to the members associated with them, and (iii) the *consistency* of the functions that relate members of different levels within the matched dimensions.

We then propose two different approaches to the problem of integration that try to enforce "good" matchings, according to the above properties. The first technique refers to a scenario of loosely coupled integration, in which we need to identify the common information between sources (intuitively, the intersection), while preserving their autonomy. With this approach drill-across queries are performed over the original sources. The goal of the second technique is rather merging the sources (intuitively, making the union) and refers to a scenario of tightly coupled integration, in which we intend to build a materialized view that includes the sources. With this approach, queries are then performed against the view built from the sources. As a preliminary tool, we introduce a powerful technique, the *chase of dimensions*, that can be used in both approaches to test for consistency and combine the content of the dimensions to integrate.

Finally, we show architecture and functionality of a system called DaWaII (Data Warehouse IntegratIon) that we have developed to test the effectiveness and efficiency of our approach. To our knowledge, DaWaII is the first CASE tool supporting the user in the integration of heterogeneous data warehouses.

1.3 Related work

The integration of heterogeneous databases has been studied in the literature extensively [11, 15, 18, 22, 27]. This problem presents, in general, many facets both at scheme and instance level (see [5, 27] for a comprehensive classification and for a survey of the solutions proposed). In this paper, we concentrate our attention on the integration of *multidimensional* data, which is clearly an instance of the general problem of database integration. Therefore, we do not address the general issues of the integration problem, such as the automatic matching of terms [13] or the resolution of structural conflicts [5]. Rather, we propose methods and techniques that take advantage of the multidimensional nature of data warehouses. It turns out that the techniques presented here are complementary to the approaches for the generic case and so, in the implementation of an effective tool, they need to be used in combination with other general integration methods. This issue will be discussed further in Sect. 6, where we will present the development of a practical integration system that incorporates some known integration approaches.

To our knowledge, the problem of data warehouse integration has never been studied in a systematic way and so this paper proposes the first approach to its solution. There are however a number of issues that are related to this problem and have been investigated in the literature.

First of all, the concept of compatibility among dimensions, which is at the basis of our work, has been discussed, under the name of "conformity", by Kimball [17] in the context of data warehouse design. However, our notion of compatibility has been introduced purposely for the goal of integration and therefore is more suitable than the notion of conformity to study how existing autonomous data marts can be combined.

An issue related to the integration of data warehouses, which has been studied in the context of statistical databases, is the *derivability* of summary data. This notion has been defined by Sato [28] as the problem of deciding whether a summary data, which is the counterpart of a fact table in a statistical database, can be inferred from another summary data organized over a different classification, that is, aggregated in a different way. The concept has been extended by Malvestuto et al., by considering the case in which the source is composed by several heterogeneous data sets [20, 21]. They proposed an algebraic approach to this problem and provided some necessary and sufficient conditions of derivability. However, both the framework and the goal of these studies is rather different from ours. Their statistical model has some similarity with a multidimensional data model but also some important diversities [29]. For instance, the notions of dimension and of aggregation hierarchy, which are important ingredients in OLAP applications, are not explicit in this model. Moreover, we follow here a constructive approach, by providing methods and algorithms that can be actually used in an integration system.

Some work has been done on the problem of integrating a data mart with external data, stored in various formats: XML [16, 25, 31] and object-oriented [24]. This is clearly related with but different from the integration of heterogeneous data marts. More specifically, this problem is related to our tightly coupled approach to integration, in that a dimension is "enriched" with data coming from another external data source, not necessarily multidimensional. On the other hand, our loosely coupled approach to integration is related to the problem of *applicability* of drill-across queries. This issue has been addressed by Abelló et al. [1] by introducing a number of relationships that can be used to navigate between different data marts of the same data warehouse. Again, this problem (as well as the context) is related but different form ours.

Finally, we mention that the chase of dimensions can be viewed as an exact method of *missing value imputation*, which has been studied in statistical data analysis and classification, for instance, by use of estimation with the EM algorithm [26].

Some of the results of this paper have been presented, in a preliminary form, in [9]. The short description of DaWaII has been done in [30] and a demo of this tool has been given in [10].

1.4 Organization of the paper

The paper is organized as follows. In Sect. 2 we recall a multidimensional model that will be used throughout the paper. In Sect. 3 we present the notion of dimension matching and provide a basic tool, called d-chase, for the management of matchings. In Sect. 4 we illustrate two techniques for dimension integration and, in Sect. 5, we describe how they can be used to integrate heterogeneous data marts. In Sect. 6 we present the practical system that we have developed to test the approach and finally, in Sect. 7, we draw some conclusions and sketch future work.

2 Preliminaries

In this section, we introduce a number of preliminary notions that are at the basis of our investigation, namely, a conceptual model for multidimensional databases and an algebra for the manipulation of dimensions. These are preceded by an introductory section to multidimensional data.

2.1 Multidimensional databases

A data warehouse is an integrated collection of operational, enterprise-wide data that is built to support decision making. Traditionally, the elements of a data warehouse that are subject of analysis are called *facts* and the specific aspects of a fact that are relevant for decision making are called *measures*. For instance, in a data warehousing application for a retail company, possible facts are the sales an the measures can be the number of units sold, the income and the cost.

It is widely recognized that the effectiveness of analysis strictly depends on the ability of describing facts in a data warehouse according to appropriate *dimensions*, that is, "perspectives" under which facts can be examined. For instance, in the above application for a retail company, it is useful to organize the sales along dimensions like products commercialized by the company, stores selling these products and days in which sales occur. Each dimension is usually structured into a hierarchy of *levels*, obtained by grouping in different ways elements of the dimension. For instance, we might be interested in grouping products by brands and categories, and days by months and years. When the members of a level l_1 can be grouped to members of another level l_2 it is often said that l_1 *rolls up* to l_2. For instance, the level "product" rolls up to the level "brand".

When a database is organized according to facts and dimensions it is often called *multidimensional*. In practical tools, a multidimensional database is usually presented to the analyst as a collection of *data cubes* (also called *data marts*) having a "physical" dimension for each "conceptual" dimension of measurement: a coordinate of the data cube specify a combination of level members and the corresponding cell contains the measure associated with such combination. If the application relies on the relational database technology, the content of a data mart is stored in a *star* schema, having a central table for the fact on which the analysis is focused, and a number of tables for the dimensions of analysis [17].

In order to make our study independent of a specific technology, we will present in the following section a conceptual multidimensional data model that includes the notions presented informally in this section and generalizes the way in which multidimensional databases are implemented in commercial systems. We have shown that this high-level representation can be easily mapped to the data models adopted by practical systems [6].

2.2 A dimensional data model

$\mathcal{MD}$ is a conceptual data model for multidimensional databases that we have introduced, in an earlier study, with a different goal [6]. As we have said in the previous section, this model generalizes the notions commonly used in multidimensional systems and, for this reason, is adopted as framework of reference for the present study. $\mathcal{MD}$ is based on two main constructs: the dimension and the data mart.

Definition 1 (Dimension) An $\mathcal{MD}$ *dimension* d is composed of:

- A *scheme* $S(d)$, made of:
  - A finite set $L = \{l_1, \dots, l_n\}$ of *levels*, and

  - A partial order $\preceq$ on L (if $l_1 \preceq l_2$ we say that l_1 *rolls up to* l_2);
- An *instance* $I(d)$, made of:
  - A function m associating *members* with levels; and
  - A family of functions ρ including a *roll up function* $\rho^{l_1 \to l_2} : m(l_1) \to m(l_2)$ for each pair of levels $l_1 \preceq l_2$.

We assume that L contains a bottom element $\perp$ (w.r.t. $\preceq$) whose members represent real world entities that we call *basic*.[1] Members of the other levels represent groups of basic members. Let $\{\tau_1, \ldots, \tau_k\}$ be a predefined set of *base types*, (including integers, real numbers, etc.).

Definition 2 (Data mart) An $\mathcal{MD}$ *data mart* f over a set D of dimensions is composed of:

- A *scheme* $f[A_1 : l_1, \ldots, A_n : l_n] \to \langle M_1 : \tau_1, \ldots, M_m : \tau_m \rangle$, where each A_i is a distinct *attribute* name, each l_i is a level of some dimension in D, each M_j is a distinct *measure* name, and each τ_j is some base type; and
- An *instance*, which is a partial function mapping coordinates for f to facts for f, where:
  - A *coordinate* is a tuple over the attributes of f mapping each attribute name A_i to a member of l_i;
  - A *fact* is a tuple over the measures of f mapping each measure name M_j to a value in the domain of type τ_j.

Example 1 Figure 1 shows a Sales data mart that represents daily sales of products in a chain of stores. This data mart is organized around four dimensions: p_1 (product), s_1 (store), m_1 (promotion) and t_1 (time). The measures of this data mart are the quantity sold, the total income and the cost of the products sold in a given day and store, and (possibly) offered in a given promotion.

In $\mathcal{MD}$, a *data warehouse* is a collection of data marts.

It is worth noting that, according to the traditional database terminology, $\mathcal{MD}$ is a *conceptual* data model, since its schemes represent real world entities. We will show that this property allows us to keep the study general and simple at the same time. It follows that $\mathcal{MD}$ schemes can be implemented using several *logical* data models [4]. An approach to the design of a data warehouse, which includes the implementation in a logical model, has been presented in [6].

2.3 An algebra for dimensions

We now introduce the *dimension algebra* (DA), a useful tool for the manipulation of existing dimensions. DA is based on three operators, as follows. Let d denote a dimension having scheme $S(d) = (L, \preceq)$ and instance $I(d) = (m, \rho)$.

[1] In [8], we called them *ground* members.

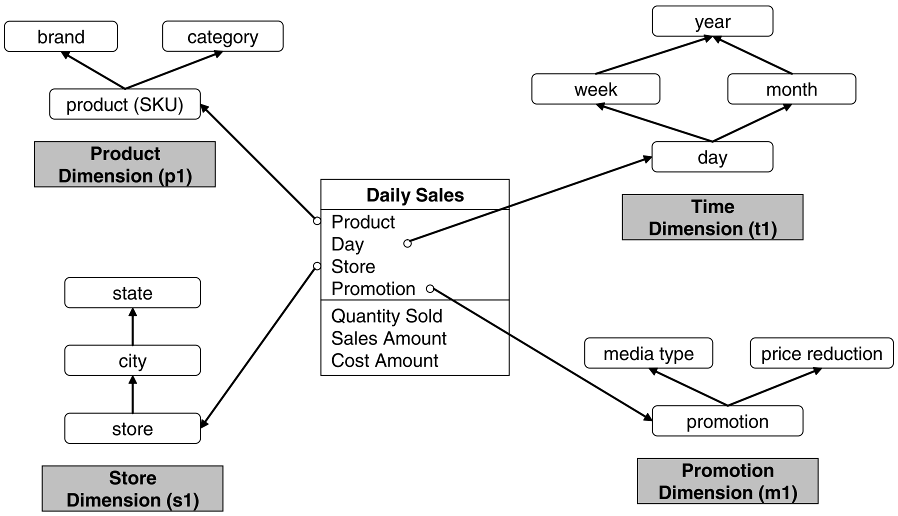


Fig. 1 Sales data mart

Definition 3 (Selection) Let G be a subset of the basic members of d. The *selection* $\sigma_G(d)$ of G over d is the dimension d' such that:

- The scheme of d' is the same of d,
- The instance of d' contains: the basic members occurring in G, the members of d that can be reached from them by applying roll-up functions in ρ, and the restriction of the roll-up functions of d to the members of d'.

Definition 4 (Projection) Let X be a subset of the levels of d that includes $\perp_d$. The *projection* $\pi_X(d)$ of d over X is the dimension d' such that:

- The scheme of d' contains X and the restriction of $\preceq$ to the levels in X,
- The instance of d' contains: the members of d that belong to levels in X and the roll-up functions $\rho^{l_1 \to l_2}$ of d involving levels in X.

Definition 5 (Aggregation) Let l be a level in L. The *aggregation* $\psi_l(d)$ of d over l is the dimension d' such that:

- The scheme of d' contains l, the levels of d to which l rolls up, and the restriction of $\preceq$ to these levels,
- The instance of d' contains: the members of d that belong to levels in d' and the roll-up functions $\rho^{l_1 \to l_2}$ of d involving levels in d'.

For a DA expression E and a dimension d, we denote by $E(d)$ the dimension obtained by applying E to d.

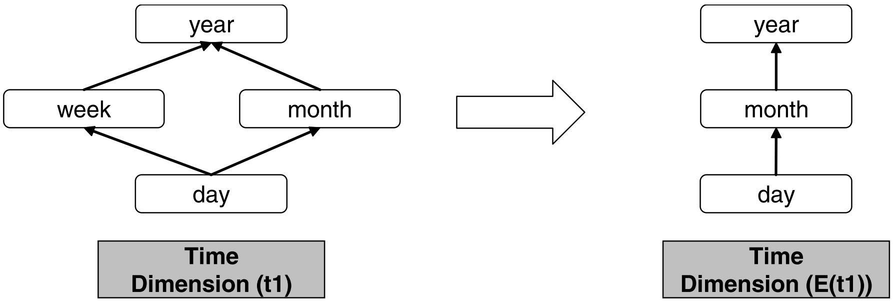


Fig. 2 Application of a DA expression

Example 2 Let us consider the time dimension t_1 of the data mart in Fig. 1 and let D_{2004} denote the days that belong to year 2004. The DA expression

$$E = \pi_{day,\ month,\ year}(\sigma_{D_{2004}}(t_1))$$

generates a new dimension with level day, month and year having as basic members all the days of 2004 (see Fig. 2).

The following is a desirable property of DA expressions.

Definition 6 (Lossless expression) A DA expression E over a dimension d is *lossless* if for each member o in $E(d)$, all the basic members that roll up to o in d, or none of them, belong to $E(d)$.

In [8] we have shown that the satisfaction of this property prevents inconsistencies between aggregations computed over d and aggregations computed over $E(d)$.

DA expressions involving only projections and aggregations are always lossless [8]. On the other hand, if a DA expression involves selections, the lossless property can fail to hold: it depends on the particular sets of elements chosen to perform the selections.

3 Matching autonomous dimensions

In this section we address the basic problem of matching heterogeneous dimension and introduce a basic procedure, called d-chase, for the management of matchings between dimensions.

In what follows, d_1 and d_2 denote two dimensions, belonging to different data marts, having scheme $S(d_1) = (L_1, \preceq_1)$ and $S(d_2) = (L_2, \preceq_2)$, and instance $I(d_1) = (m_1, \rho_1)$ and $I(d_2) = (m_2, \rho_2)$, respectively.

3.1 Dimension matching and its properties

Let us start with the basic notion of dimension matching.

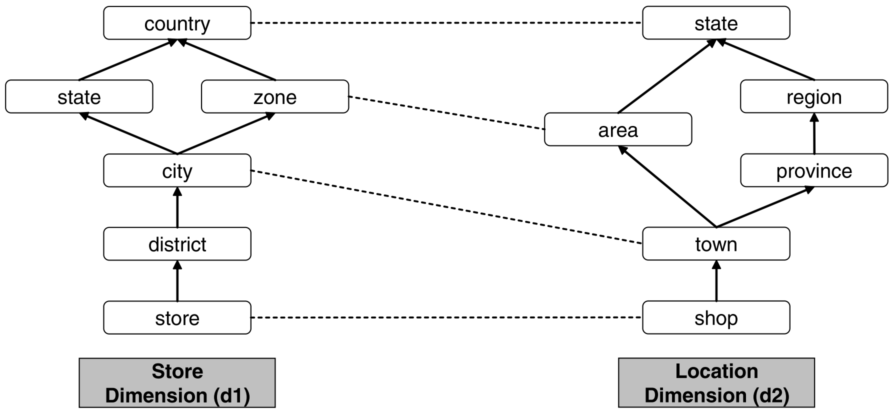


Fig. 3 A matching between two dimensions

Definition 7 (Dimension Matching) A *matching* between two dimensions d_1 and d_2 is a (one-to-one) injective partial mapping μ between L_1 and L_2.

With a little abuse of notation, given a matching μ, we will denote by μ also its inverse. We also extend μ to sets of levels in the natural way (that is, $\mu(L)$ is the set containing $\mu(l)$ for each level l in L). Also, we will assume that μ is the identity on the levels on which it is not defined.

Example 3 Figure 3 shows an example of matching between two geographical dimensions that associates store with shop, city with town, zone with area, and country with state.

As we have said in the Introduction, we will not address the problem of the automatic derivation of a dimension matching, since this is outside the goal of this paper. However, standard techniques for semantic reconciliation [27] can be used for this purpose. Actually, we have implemented some of them in our system, which is capable to suggest possible matchings according to both a top-down and a bottom-up strategy, as described in more detail in Sect. 6.

A number of desirable properties can be defined over a matching between dimensions. Intuitively, we say that a matching is: (i) *coherent* if it preserves the hierarchy defined on their levels, (ii) *sound* if it preserves level membership, and (iii) *consistent* if it preserves the roll-up relationships between members of different levels. More precisely, we have the following definition.

Definition 8 (Matching Properties) Let μ be a matching between two dimensions d_1 and d_2. Then:

- **Coherence:** μ is *coherent* if, for each pair of levels l, l' of d_1 on which μ is defined, $l \preceq_1 l'$ if and only if $\mu(l) \preceq_2 \mu(l')$;
- **Soundness:** μ is *sound* if, for each level $l \in L_1$ on which μ is defined, $m_1(l) = m_2(\mu(l))$;

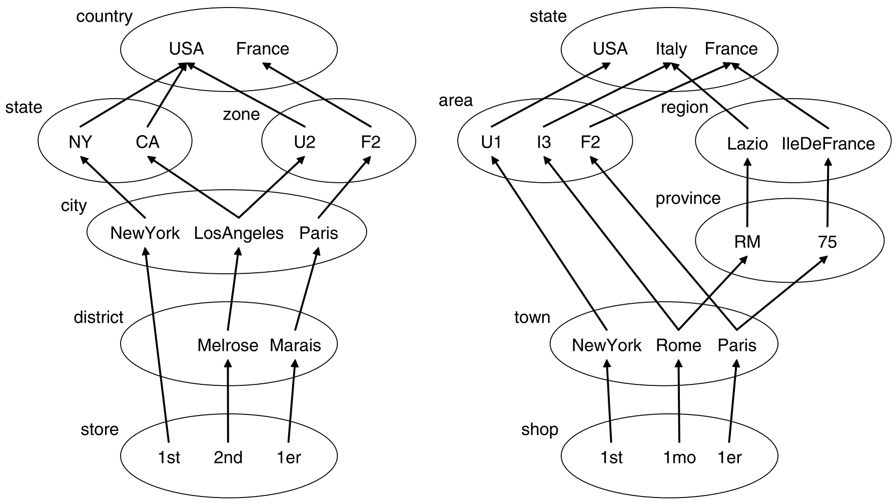


Fig. 4 Possible instances of the dimensions in Fig. 3

- **Consistency:** μ is *consistent* if, for each pair of levels $l \preceq_1 l'$ of d_1 on which μ is defined, $\rho_1^{l \to l'} = \rho_2^{\mu(l) \to \mu(l')}$.

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

Example 4 The matching in Fig. 3 is clearly coherent since, roughly speaking, no correspondence intersects another. Now assume that levels of the two dimensions be populated by the members reported in Fig. 4. Then the matching is consistent, since the roll-up functions are not contradictory, but it is not sound, since there are members in one dimension that do not appear in the other.

Note that, for sake of simplicity, soundness refers, in the theoretical investigation, to a *conceptual* view, under which two levels coincides if they are populated by the same real world entities. This choice allows us to take apart the problem of using different values to represent the same entity (or, similarly, the same value for different entities). This is clearly an important issue but, as we have said in the Introduction, is outside the scope of the paper. In Sect. 6 we will show that, in the development of a practical system, we have followed a logical approach, based on standard techniques supporting the derivation of mappings between identifiers representing the same entity.

Clearly, a perfect matching is very difficult to achieve in practice. In many cases however, autonomous dimensions actually share some information. To identify this common information, we need the ability to select a portion of a dimension. This comment leads to the following definition.

Definition 9 (Dimension Compatibility) Two dimensions d_1 and d_2 are *μ-compatible* if there exist two lossless DA expressions E_1 and E_2 over d_1 and d_2, respectively, such that μ is a perfect matching between $E_1(d_1)$ and $E_2(d_2)$.

The rationale underlying the definition of compatibility is that: (i) two dimensions may have common information; (ii) the intersection can be identified by DA expressions; and (iii) lossless expressions guarantee the correctness of OLAP operations over the intersection [8].

Example 5 The dimensional matching reported in Fig. 3 can be made perfect by applying, for instance, the following expressions to d_1 and d_2, respectively:

$$\pi_{store,\ city,\ zone,\ country}(\sigma_{\{1er\}}(d_1)),\ \pi_{shop,\ town,\ area,\ state}(\sigma_{\{1er\}}(d_2)).$$

3.2 Chase of dimensions

We now describe a procedure called d-chase (for chase of dimensions) that applies to members of autonomous dimensions and show that it can be used for integration purposes.

Let V be a set of *variables* and $L = l_1, \dots, l_k$ be a set of levels. A *tableau* T over L is a set of tuples t mapping each level l_i to a member of l_i or a variable in V.

Now, let μ be a matching between two dimensions d_1 and d_2.

Definition 10 (Matching Tableau) The *matching tableau* over d_1, d_2 and μ, denoted by $T_\mu[d_1, d_2]$, is a tableau over $L = L_1 \cup \mu(L_2)$ having a tuple t_m for each member m of a level $l \in L$ such that:

- $t_m[l] = m$,
- $t_m[l'] = \rho^{l \to l'}(m)$, for each level l' to which l rolls up,
- $t_m[l''] = v$, where v is a variable not occurring elsewhere, for all other levels in L.

Example 6 Let us consider the dimension matching in Fig. 3 and assume that levels of these two dimensions be populated by the members reported in Fig. 4. The corresponding matching tableau is the following.

Store	City	Zone	Country	District	State	Prov.	Region
1st	NewYork	v_1	USA	v_2	NY	v_3	v_4
2nd	LosAng.	U2	USA	Melrose	CA	v_5	v_6
1er	Paris	F2	France	Marais	v_7	v_8	v_9
1mo	Rome	I3	Italy	v_{10}	v_{11}	RM	Lazio
1st	NewYork	U1	USA	v_{12}	v_{13}	v_{14}	v_{15}
1er	Paris	F2	France	v_{16}	v_{17}	75	IledeFr

In this example, the first three tuples represent members of d_1 and the others members of d_2. The first four columns represent the matched levels and the other columns represent levels of the two dimensions that have not been matched. Note that a variable occurring in a tableau may represents an unknown value (for instance, in the first row, the zone in which the store *1st* is located, an information not available in the instance of d_1) or an inapplicable value (for instance, in the last row, the district in which the store *1er* is located, a level not present in the scheme of d_2).

We point out that this tableau can be easily built, in an automatic way, by just extending, with fresh labelled nulls, the members of the original dimensions (which, in relational systems, are usually just tuples in dimension tables) to the levels not occurring in them.

The *d-chase* (chase of dimensions) is a procedure inspired by an analogous procedure used for reasoning about dependencies in the relational model [2]. This procedure takes as input a tableau T over a set of levels L and generates another tableau that, if possible, satisfies a set of roll-up functions ρ defined over the levels in L. This procedure modifies values in the tableau, by applying *chase steps*. A chase step applies when there are two tuples t_1 and t_2 in T such that $t_1[l] = t_2[l]$ and $t_1[l'] \neq t_2[l']$ for some roll up function $\rho^{l \to l'} \in \rho$ and modifies the l'-values of t_1 and t_2 as follows: if one of them is a constant and the other is a variable then the variable is changed (is *promoted*) to the constant, otherwise the values are equated. If a chase step tries to identify two constants, then we say that the d-chase encounters a *contradiction*, and the process stops generating a special tableau that we denote by T_∞ and call the inconsistent tableau.

Definition 11 (D-chase) The d-chase of a tableau T, denoted by $\textit{DCHASE}_\rho(T)$, is a tableau obtained from T and a set of roll-up functions ρ by applying all valid chase steps exhaustively to T.

Example 7 By applying the d-chase procedure to the matching tableau of Example 6 we do not encounter contradictions and obtain the following tableau.

Store	City	Zone	Country	District	State	Prov.	Region
1st	NewYork	U1	USA	v_2	NY	v_3	v_4
2nd	LosAng.	U2	USA	Melrose	CA	v_5	v_6
1er	Paris	F2	France	Marais	v_7	75	IledeFr
1mo	Rome	I3	Italy	v_{10}	v_{11}	RM	Lazio

The d-chase promotes, for instance, v_1 to *U1*, and v_8 to 75.

Note that in the d-chase procedure, a promotion of a variable always corresponds to the detection of an information present in the other dimension and consistent with the available information but not previously known.

An important result about the d-chase, which just extend a known result of relational theory [2], is the following.

Lemma 1 *The d-chase process terminates on any input with a unique end result.*

Proof The d-chase is a variant of the chase procedure, a theoretical tool introduced to study the implication of constraints [3, 19] and later extended to solve other problems related to the management of relational databases [2]. In particular, the matching tableau can be viewed as a *representative instance* over a relation scheme that includes a table for each dimension. A representative instance is a tableau that includes all the tuples of the tables occurring in a relational database instance [14]. Under this view, roll-up functions correspond to functional dependencies over such scheme. The steps of the d-chase are therefore equivalent to chase steps based on functional dependencies of a standard chase procedure. One of the main result that has been proved for the chase is that it satisfies the Church-Rosser property [2], that is, the termination is guaranteed on any input and the final result is unique up to a renaming of variables. □

The following result states that the d-chase provides an effective way to test for consistency. The proof is based on a reduction of the problem to the problem of testing the consistency of a relational database against a set of functional dependencies, for which correctness results of such procedure are known [2].

Theorem 1 *A matching μ between two dimensions d_1 and d_2 is consistent if and only if* $\textit{DCHASE}_{\rho_1 \cup \mu(\rho_2)}(T_\mu[d_1, d_1]) \neq T_\infty$.

Proof Under the interpretation of the d-chase given in the proof of Lemma 1, the application of this procedure to the tableau $T_\mu[d_1, d_1]$ corresponds to the application of the standard chase procedure on a representative instance I_R built from a relational database R that includes a relation R_1 with a tuple for each basic member of d_1 and a relation R_2 with a tuple for each basic member of d_2. Also, each roll-up function $\rho^{l \to l'}$ in $\rho_1 \cup \mu(\rho_2)$ corresponds to a functional dependency $X_l \to X_{l'}$ over I_R. By a result in [14], the chase of I_R terminates without encountering contradictions if and only if I_R *globally* satisfies the functional dependencies defined for it. This means that, for each functional dependency $X_l \to X_{l'}$ and each pair of tuples t_1 originating from R_1 and t_2 originating from R_2 such that $t_1[X_l] = t_2[X_l]$, it is the case that $t_1[X_{l'}] = t_2[X_{l'}]$. Since R_1 and R_2 represent the two dimensions d_1 and d_2 and the functional dependencies the corresponding roll-up functions, this implies that the roll-up functions of the two dimensions coincide. Therefore, we have that we do not encounter contradictions by applying the d-chase to $T_\mu[d_1, d_1]$ if and only if $\rho_1^{l \to l'} = \rho_2^{\mu(l) \to \mu(l')}$, that is, if and only if μ is consistent. □

We finally define a special operation over a tableau that will be used in the following. Let T be a tableau over a set of levels L and $S = (L', \preceq)$ be the scheme of a dimension such that $L' \subseteq L$.

Definition 12 (Total projection) The *total projection* of T over S, denoted by $\pi_S^{\downarrow}(T)$, is an instance (m, ρ) of S defined as follows.

- For each level $l \in L$, $m(l)$ includes all the members occurring in the l-column of T.
- For each pair of levels l_1, l_2 in L such that $l_1 \preceq l_2$ and for each tuple t of T such that: (i) both the l_1-value and the l_2-value are defined, and (ii) there is no other tuple t' in T such that $t[l_1] = t'[l_1]$ and $t[l_2] \neq t'[l_2]$, then $\rho^{l_1 \to l_2}(t[l_1]) = t[l_2]$ and is undefined otherwise.

Example 8 The total projection of the chased tableau of Example 7 over dimension d_1 contains all the basic members that appear in the first six columns of the tableau.

Let d be a dimension and μ be a matching between d and any other dimension d', and assume that $\mathit{DCHASE}_{\rho_1 \cup \mu(\rho_2)}(T_\mu[d_1, d_1]) \neq T_\infty$. We can easily show the following.

Lemma 2 $I(d) \subseteq \pi^{\downarrow}_{S(d)}(\mathit{DCHASE}_{\rho \cup \mu(\rho')}(T_\mu[d, d']))$.

Proof The fact that each member and each roll-up function of $I(d)$ is also in the total projection over $S(d)$ of $\mathit{DCHASE}_{\rho_1 \cup \mu(\rho_2)}(T_\mu[d_1, d_1])$ easily follows from the definitions of matching tableau and of d-chase. Moreover, Example 8 illustrates a case in which the containment is strict. For instance, the total projection over d_1 contains for instance the member *1mo* that was not in the original instance of d_1. □

This result states an interesting property of the chase that goes beyond the test of consistency. If we apply the d-chase procedure over a matching tableau that involves a dimension d and then project the result over the scheme of d, we obtain the original instance and, possibly, some additional (and consistent) information that has been identified in the other dimension.

4 Two approaches to dimension integration

In this section we propose two different approaches to the problem of the integration of autonomous data marts that rely on the issues introduced in the previous section. As we have said in the Introduction, the first approach refers to a scenario of *loosely coupled* integration, in which we need to identify the common information between sources and perform drill-across queries over the original sources. The second refers to a scenario of *tightly coupled* integration, in which we want to build a view that combine the different information sources and perform queries over this view. The d-chase procedure presented in the previous section will be used in both approaches.

4.1 A loosely coupled approach

In a loosely coupled integration scenario the goal is to select data that is shared between the sources. Thus, given a pair of dimensions d_1 and d_2 and a matching μ between them, the approach aims at deriving two expressions that makes μ perfect. The approach is based on Algorithm 1, which is reported in Fig. 5 and is structured as follows.

- First of all, the algorithm selects the levels L of d_1 involved in the matching μ (Step 1). Then, for each minimal level l_m in L (that is, for which there is no other level $l \in L$ such that $l \preceq_1 l_m$), it selects only the levels to which l_m rolls up (Step 3). The rationale is to find the expressions that detect the intersection of d_1 and d_2 in the levels above l_m. If there are several minimal levels, the algorithm iterates over all of them (Step 2) thus possibly generating several pairs of expressions.
- Step 4 consists of a test for the coherence of μ according to Definition 8. Actually, this test can be done efficiently by taking advantage of the transitivity of $\preceq$.
- In Step 5 two preliminary expressions E_1 and E_2 are identified: they aggregate over l_m ($\mu(l_m)$) and project over L ($\mu(L)$).

```
Algorithm 1
Input: two dimensions $d_1$ and $d_2$ and a matching $\mu$;
Output: two expressions $E_1$ and $E_2$ that make $\mu$ perfect;
begin
(1)  $L :=$ the levels of $d_1$ involved in $\mu$;
(2)  for each minimal level $l_m$ of $L$ do
(3)      $L := L - \{l \in L$ such that $l_m \not\preceq_1 l\}$;
(4)      if there exist $l_1, l_2 \in L$ such that
             $l_1 \preceq_1 l_2$ and $\mu(l_1) \not\preceq_2 \mu(l_2)$
         then output 'not coherent' and exit;
(5)      $E_1 := \pi_L(\psi_{l_m}(d_1))$; $E_2 := \pi_{\mu(L)}(\psi_{\mu(l_m)}(d_2))$;
(6)      $M := m_1(l_m) \cap m_2(\mu(l_m))$;
(7)      $T := T_\mu[\sigma_M(E_1(d_1)), \sigma_M(E_2(d_2))]$;
(8)      $T := DCHASE_{\rho_1 \cup \mu(\rho_2)}(T)$;
(9)      if $T = T_\infty$ then output 'not consistent' and exit;
(10)     $d_1 := \pi^{\downarrow}_{S(d_1)}(T)$; $d_2 := \pi^{\downarrow}_{S(d_2)}(T)$;
(11)     for each non basic member $m \in m_{1,2}(l)$ in $T$ do
(12)         if $\exists$ a basic $m' \in m_{1,2}(l')$ such that $l' \preceq_{1,2} l$ and
                 $\rho_{1,2}^{l' \to l}(m') = m$ and $m'$ does not occur in $T$
             then $T := T - \{t \mid t[l] = m\}$
(13)     $M := \{m \mid t[l_m] = m$ for some $t \in T\}$;
(14)     $E_1 := \sigma_M(E_1(d_1))$; $E_2 := \sigma_M(E_2(d_2))$;
(15)     output $E_1$ and $E_2$;
     endfor
end
```

Fig. 5 An algorithm for deriving the common information between two dimensions

- The rest of the algorithm aims at finding the selection that, applied to E_1 and E_2, identifies only the common members in the two dimensions. This is done by building a matching tableau over the members that occur both in l_m and $\mu(l_m)$ (Steps 6 and 7) and then chasing it (Step 8). This serves to test for consistency on the restriction of μ to the levels in L.
- Step 10 serves to guarantees the soundness of the matching: by projecting the result of the d-chase over the original dimensions we are sure that there are not members of one dimensions that do not appear in the other dimension.
- Steps 11 and 12 serves to identify, from the members occurring in the working tableau T, all the members that invalidate the property of lossless expression (Definition 6).
- Finally, all the members that still occur in T at level l_m are used to perform the final selection (Steps 13 and 14).

Example 9 Let us consider the application of Algorithm 1 to the dimensions and the matching in Fig. 3, assuming that the dimensions are populated by the members reported in Fig. 3. Since the matching involves the bottom levels of the two dimensions, no aggregation is required and the first part of the algorithm generates the following expressions: $\pi_{store,\ city,\ region,\ country}(d_1)$ and $\pi_{shop,\ town,\ area,\ state}(d_2)$. The intersection of the basic members contains only the stores *1st* and *1er* and so the d-chase produces the following tableau:

Store	City	Zone	Country	District	State	Prov.	Region
1st	NewYork	U1	USA	v_2	NY	v_3	v_4
1er	Paris	F2	France	Marais	v_7	75	IledeFr

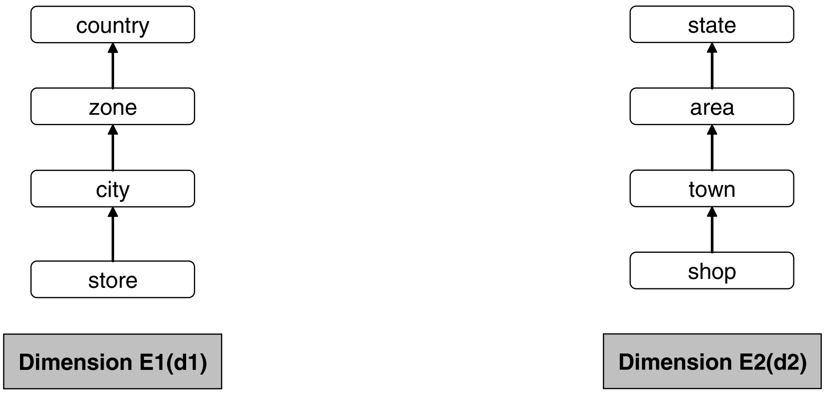


Fig. 6 The dimensions generated by Algorithm 1 on the matching in Fig. 3

This tableau contains, at the country level, the member USA to which the member 2nd of d_2 rolls up, but 2nd is not present in the tableau. It follows that in Step 12 the first row is deleted and we obtain as output of the algorithm the following final expressions:

$$\sigma_{\{1er\}}(\pi_{store,\ city,\ region,\ country}(d_1)),$$
$$\sigma_{\{1er\}}(\pi_{shop,\ town,\ area,\ state}(d_2)).$$

The schemes of the dimensions we obtain by applying these expressions to the original dimensions are reported in Fig. 6.

On the basis of to the results of the previous section, we can state the following correctness result. The proof proceed by construction, showing that the various step of the algorithm guarantee, according to the basic results provided in Sect. 3, the satisfaction of the properties that make perfect a dimension matching. Namely, its consistency, soundness and coherence.

Theorem 2 *The execution of Algorithm* 1 *over two dimensions d_1 and d_2 and a matching μ between them returns two expressions E_1 and E_2 that make μ perfect if and only if d_1 and d_2 are μ-compatible.*

Proof (If) If d_1 and d_2 are μ-compatible then by definition there exist two expressions E_1 and E_2 that make μ perfect. By the coherence of μ over $E_1(d_1)$ and $E_1(d_1)$ it follows that there is not a pair of levels $l_1, l_2 \in L$ such that $l_1 \preceq_1 l_2$ and $\mu(l_1) \npreceq_2 \mu(l_2)$, and therefore Algorithm 1 does not exit at Step 4. Moreover, by the consistency of μ and by Theorem 1 it follows that the matching tableau over d_1, d_2 and μ does not encounter contradictions, and so Algorithm 1 does not exit at Step 4. Hence, Algorithm 1 terminates and correctly returns two expressions.
(Only if) Let E_1 and E_2 the expressions returned by the execution of the Algorithm 1 over d_1, d_2 and μ. As discussed above, Step 4 of the algorithm consists of testing

for coherence of μ according to Definition 8. Step 5 generates two preliminary expressions E_1 and E_2 that aggregate over l_m ($\mu(l_m)$) and project over L ($\mu(L)$). Since no selection is involved in these expressions, by a result in [8], they are lossless. In Steps 6 and 7 the algorithm builds a matching tableau over the members that occur both in l_m and $\mu(l_m)$ and, in Step 8, it applies the d-chase procedure. Then, by Theorem 1, Step 9 corresponds to a test of consistency for the restriction of μ to the levels in L. By Lemma 2, it follows that the d-chase can identify new members of a level of one dimension in the corresponding level of other dimension. To preserve soundness of μ, the total projections performed by Step 10 guarantees that the matched levels have the same members. All the members that occur in the working tableau T and that can invalidate the lossless property of the expression are eliminated in Steps 11 and 12, according to Definition 6. Finally, Steps 13 and 14 perform the final selection of the members that still occur in T. Since the properties of coherence, soundness, and consistency of μ and of the lossless of E_1 and E_2 have been guaranteed in the construction of these expressions, it follows that μ is a perfect matching between $E_1(d_1)$ and $E_2(d_2)$, and therefore d_1 and d_2 are μ-compatible. □

The most expensive step of the algorithm is the d-chase that requires time polynomial with respect to the size of the tableau, which in turn depends on the cardinality of the dimensions involved. It should be said however that the size of dimensions in a data warehouse is much smaller than the size of the facts. Moreover, the content of a dimension is usually stable in time. It follows that the algorithm can be executed off-line and occasionally, when it arises the need for integration or when changes on dimensions occur.

4.2 A tightly coupled approach

In a *tightly coupled integration*, we want to build a materialized view combining different data sources and perform queries over this view. Our goal is the derivation of new dimensions obtained by merging the dimensions of the original data sources. In this case, given a pair of dimensions d_1 and d_2 and a matching μ between them, the integration technique aims at deriving a new dimension obtained by merging the levels involved in μ and including, but taking apart, all the other levels. The approach is based on Algorithm 2, which is reported in Fig. 7 and consists of phases.

- Similarly to Algorithm 1, Algorithm 2 performs first a check for coherence of the input matching.
- If the test of coherence is successful, it builds a new (preliminary) dimension scheme $S = (L, \preceq)$ by merging the levels (Step 3) and the roll-up relations between levels (Step 4) of the input dimensions (the transitive closure serves to guarantee that the relation we obtain is a partial order).
- The next step takes into account the special case in which the relation $\preceq$ we obtain has more than one minimal level. In this case, in Steps 6 and 7, two new auxiliary bottom levels $\perp_1'$ and $\perp_2'$ are added below the original bottom levels $\perp_1$ and $\perp_2$ of d_1 and d_2, respectively, and populated by the same members. Then, the map $(\perp_1', \perp_2')$ is added to μ (Step 8) and the scheme $S = (L, \preceq)$ is modified accordingly (Steps 9 and 10).

```
Algorithm 2
Input: two dimensions $d_1$ and $d_2$ and a matching $\mu$;
Output: a new dimension $d$ that embeds $d_1$ and $d_2$;
begin
(1) $L :=$ the levels of $d_1$ involved in $\mu$;
(2) if there exist $l_1, l_2 \in L$ such that
        $l_1 \preceq_1 l_2$ and $\mu(l_1) \not\preceq_2 \mu(l_2)$
    then output 'not coherent' and exit;
(3) $L := L_1 \cup \mu(L_2)$;
(4) $\preceq := (\preceq_1 \cup \mu(\preceq_2))^+$;
(5) if $\preceq$ has several minimal levels
    then
(6)    $d'_1 := d_1$ augmented with a new bottom level $\bot'_1$;
(7)    $d'_2 := d_2$ augmented with a new bottom level $\bot'_2$;
(8)    $\mu' := \mu \cup \{(\bot'_1, \bot'_2)\}$;
(9)    $L := L \cup \bot'_1$;
(10)   $\preceq := (\preceq'_1 \cup \mu(\preceq'_2))^+$;
    else $d'_1 := d_1$; $d'_2 := d_2$; $\mu' := \mu$;
(11) $T := DCHASE_{\rho'_1 \cup \mu(\rho'_2)}(T_{\mu'}[d'_1, d'_2])$;
(12) if $T = T_\infty$ then output 'not consistent' and exit;
(13) $d := \pi^{\downarrow}_{(L,\preceq)}(T)$;
(14) output the dimension $d$;
end
```

Fig. 7 An algorithm for merging two dimensions

- A matching tableau is then built on the (possibly modified) dimensions and a d-chase procedure is applied to the tableau (Step 11).
- Finally, if no contradiction is encountered in the d-chase, the total projection of the resulting tableau over the scheme S generates the output dimensions (Steps 12 and 13).

Example 10 Let us consider again the matching between dimensions in Fig. 3 but assume that the level store does not map to the level shop. This means that the corresponding concepts are not related. It follows that the union of the schemes of the two dimensions produces two minimal levels. Then, the application of Algorithm 2 to this matching introduces two bottom levels below store and shop. The scheme of the dimension generated by the algorithm is reported in Fig. 8. If the dimensions are populated by the members in Fig. 3, the output instance contains all the members occurring in the chased tableau reported in Example 7.

We say that a dimension d *embeds* another dimension d' if there exists a DA expression E such that $E(d) = d'$. On the basis of the discussion above, we can state the following result. Also in this case, the proof proceed by construction, showing that the various step of the algorithm guarantee, according to the basic results provided in the previous sections, that the dimension generated by the algorithm correctly includes the original dimensions and satisfy the basic properties of consistency and coherence.

Theorem 3 *The execution of Algorithm* 2 *over two dimensions d_1 and d_2 and a coherent and consistent matching μ between them returns a new dimension d embedding both d_1 and d_2.*

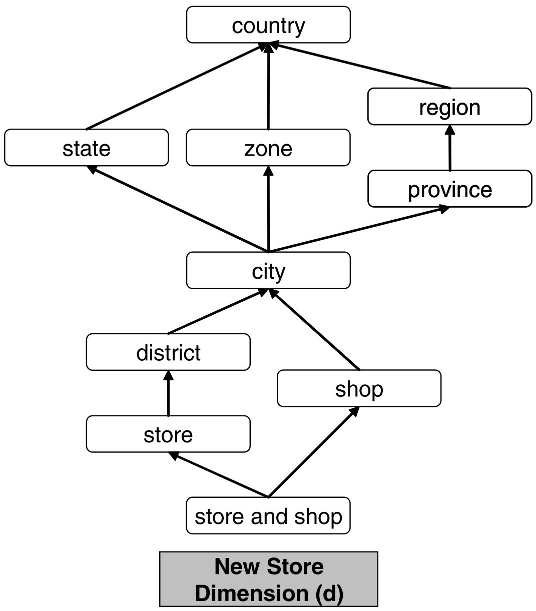


Fig. 8 The dimension generated by Algorithm 2 on a variant of the matching in Fig. 3

Proof In Step 2, the algorithm performs a check for coherence of the input matching. If the test is successful, Steps 3 and 4 build a new dimension scheme $S = (L, \preceq)$ by merging the levels and the roll-up relations between levels of the input dimensions. For the latter, we need to guarantee that the relation we obtain is a partial order. Irreflexivity and asymmetry follow by the coherence of the matching. To enforce transitivity, the transitive closure is computed over the union of the two roll-up relations. Assume that the relation $\preceq$ we obtain in this way has more than one minimal level. In this case, in Steps 6 and 7, two new auxiliary bottom levels $\bot'_1$ and $\bot'_2$ are added below the original bottom levels $\bot_1$ and $\bot_2$ of d_1 and d_2, respectively. In order to guarantee the uniqueness of the bottom level for L without generating undesirable inconsistencies, these levels are populated with copies of the basic members of $\bot_1$ and $\bot_2$, suitably renamed so that the intersection of the two sets of copies is empty. Then, two new roll-up functions $\rho^{\bot'_1 \to \bot_1}$ and $\rho^{\bot'_2 \to \bot_2}$ mapping each copy to the corresponding member are added to the instances of the dimensions. Step 8 adds the map $(\bot'_1, \bot'_2)$ to μ and Steps 9 and 10 modify the scheme $S = (L, \preceq)$ accordingly. By Theorem 1 Step 11 corresponds to a test for consistency, and Steps 12 and 13 perform the total projection of the resulting tableau over the scheme S. By Lemma 2 and by construction, it follows that d embeds both d_1 and d_2. □

Again, the complexity of the algorithm is bounded by the d-chase procedure that requires polynomial time in the size of the dimensions involved. Hence, we can make for this algorithm the same considerations done for Algorithm 1.

5 Data mart integration

Drill-across queries have the goal of combining and correlating data from multiple data marts, and are especially useful to perform value chain analysis [17]. These queries are based on joining different data marts over common dimensions. Since join operations combine relations on the basis of *common* data, the existence of shared information between data marts is needed in order to obtain meaningful results. In this section we discuss how the techniques described in Sect. 4 can be used to perform this kind of queries.

5.1 An algebra for data marts

In order to show how data marts can be manipulated, we will use an abstract algebra for data marts, called DMA, that we have introduced in an earlier paper [7]. This allows us to keep the approach general and independent of a specific technology.

DMA is based on a number of operators that apply to $\mathcal{MD}$ data marts and produce an $\mathcal{MD}$ data mart as result. In this way, the various operators can be composed to form the *expressions* of the language. We do not present all of them here, but just recall those that will be used in the examples that follow.

- **Join**: if f and f' are data marts having schemes $f[A_1, \ldots, A_i, \ldots, A_n] \to \langle M_1, \ldots, M_m \rangle$ and $f'[A'_1, \ldots, A'_j, \ldots, A'_{n'}] \to \langle M'_1, \ldots, M'_{m'} \rangle$, respectively, then $f \bowtie_{A_i = A'_j} f'$ is a data mart f'' having scheme $f''[A_1, \ldots, A_n, A'_1, \ldots, A'_{n'}] \to \langle M_1, \ldots, M_m, M'_1, \ldots, M'_{m'} \rangle$. This data mart contains a tuple τ'' for each pair of tuples $\tau \in f$ and $\tau' \in f'$ such that $\tau[A_i] = \tau[A'_j]$; the tuple τ'' coincides with τ on the attributes and measures of f and with τ' on the attributes and measures of f'.
- **Roll up**: If f is a data mart having scheme $f[A_1, \ldots, A_i, \ldots, A_n] \to \langle M_1, \ldots, M_m \rangle$ and A' is an new attribute over a level l', such that the level of A_i rolls up to l', then $\varrho^{A':l'}_{A_i:l_i}(f)$ is a data mart f' having scheme $f'[A_1, \ldots, A_i, \ldots, A_n, A'] \to \langle M_1, \ldots, M_m \rangle$. This data mart contains a tuple τ' for each tuple τ of f obtained by extending τ with a value o for A', where o is the value to which $\tau[A_i]$ rolls up.
- **Aggregation**: If f is a data mart having scheme $f[A_1, \ldots, A_i, \ldots, A_n] \to \langle M_1, \ldots, M_m \rangle$ and $g_1, \ldots, g_m$ are aggregate functions, then $\psi^{N_1 = g_1(M_1), \ldots, N_m = g_m(M_m)}_{A_1, \ldots, A_i}(E)$ is a data mart having scheme $f'[A_1, \ldots, A_i] \to \langle N_1, \ldots, N_m \rangle$. This data mart contains a set of tuples obtained by first grouping over the attributes $A_1, \ldots, A_i$ the tuples of f, an then applying to each group the aggregate functions $g_1, \ldots, g_m$ to the measures $M_1, \ldots, M_m$, respectively.

DMA also includes selection and projection operators for data marts, an operator to apply scalar functions to measures, and an operator to transform measures into attributes and vice versa [7].

5.2 Loose integration of data marts

Let us consider the example in Fig. 9, in which we wish to integrate a sales data mart with a data mart storing weather information, in order to correlate sales of products with weather conditions. As indicated in the figure, the integration between these data

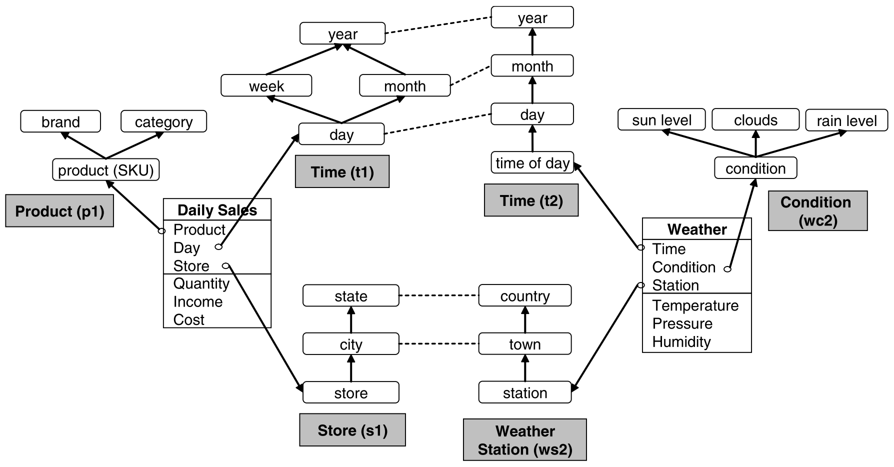


Fig. 9 Integration of Sales and Weather Station data marts

sources can be based on the matchings between the time dimensions (t_1 and t_2) at day level (since information on sales at a specific time are not available), and the location dimensions (s_1 and ws_2) at the city level (to correctly obtain the whether in a given store location).

In a loose integration of these data marts, we first identify the intersection between these pairs of dimensions using the algorithm proposed in Sect. 4.1 and then generate an integrated view obtained by joining the original data marts. Actually, the algorithm also checks for the *quality* of such intersection, according to the property of dimension compatibility, which guarantees the correctness of aggregate operations over the integrated view, as discussed in Sect. 3.

The application of Algorithm 1 to this input returns two pairs of expressions. The first pair allows the generation, from the original time dimensions, of two identical dimensions containing the shared information:

$$\widehat{t_1} = \pi_{day,\ month,\ year}(\sigma_{day_{t_1} \cap day_{t_2}}(t_1)),\ \widehat{t_2} = \psi_{day}(\sigma_{day_{t_1} \cap day_{t_2}}(t_2)),$$

where $\mathrm{day}_{t_1} \cap \mathrm{day}_{t_2}$ denotes the days in common that make the matching between t_1 and t_2 perfect.

The second pair does the same for the location dimensions:

$$\widehat{l_1} = \psi_{city}(\sigma_{city_{s_1} \cap city_{ws_2}}(s_1)), \qquad \widehat{l_2} = \psi_{city}(\sigma_{city_{s_1} \cap city_{ws_2}}(ws_2)),$$

where $\mathrm{city}_{s_1} \cap \mathrm{city}_{ws_2}$ denotes the cities in common that make the matching between s_1 and ws_2 perfect.

It turns out that we can join the two data marts to extract daily and city-based data, but hourly or store-based data can not be computed. Moreover, if we apply the above expressions to the underlying dimensions before executing the drill-across operation we prevent inconsistencies in subsequent aggregations over the result of the join.

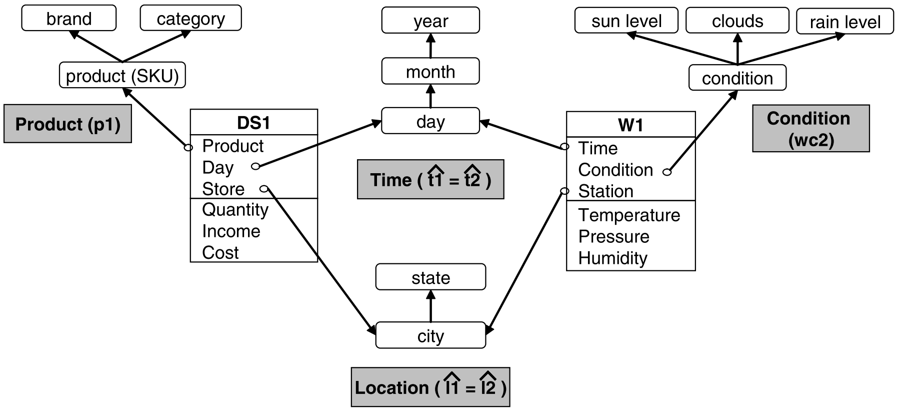


Fig. 10 The integration based on the common portions of the dimensions

It follows that, to build an integrated view, we need to apply the following DMA expressions over the original data marts:

$$DS_1 = \psi^{Q'=sum(Q),I'=sum(I)s,C'=sum(C)}_{P,D,S'}(\varrho^{S':city}_{S:store}(Sales)),$$

$$W_1 = \psi^{Tp'=avg(Tp),P'=avg(P)s,H'=avg(H)}_{T',C,WS'}(\varrho^{WS':town}_{WS:station}(\varrho^{T':day}_{T:timeOfDay}(Weather))).$$

Finally, these data marts can be joined over the new dimensions with the following expression:

$$DS_1 \bowtie_{\widehat{t_1}.day=\widehat{t_2}.day,\ \widehat{l_1}.city=\widehat{l_2}.town} W_1$$

and we obtain the view shown in Fig. 10. Drill-across queries can now be performed over this view. The above definitions suggest how translate queries against the view to queries over the original sources.

5.3 Tight integration of data marts

The tightly coupled approach aims at combining data from different dimensions, intuitively, by computing their union rather than their intersection. This can be useful when we need to reconcile and merge two data marts that have been developed independently.

Consider again the example in Fig. 9 and assume that now we want to follow this approach. By applying Algorithm 2 over the time and location dimensions, we generate two new dimensions reported in Fig. 11. Note that, the location dimension that we have obtained contains a new bottom level, which includes all the stores and the whether stations. These dimensions can be materialized and used for both data marts in the place of the original ones. We can then refer to the homogeneous integrated scenario reported in Fig. 11 and perform drill-across queries over the shared dimensions.

Note that, similarly to the approach illustrated in the previous section, to perform a join operation we first need to: (i) aggregate the Wheatear data mart at the day

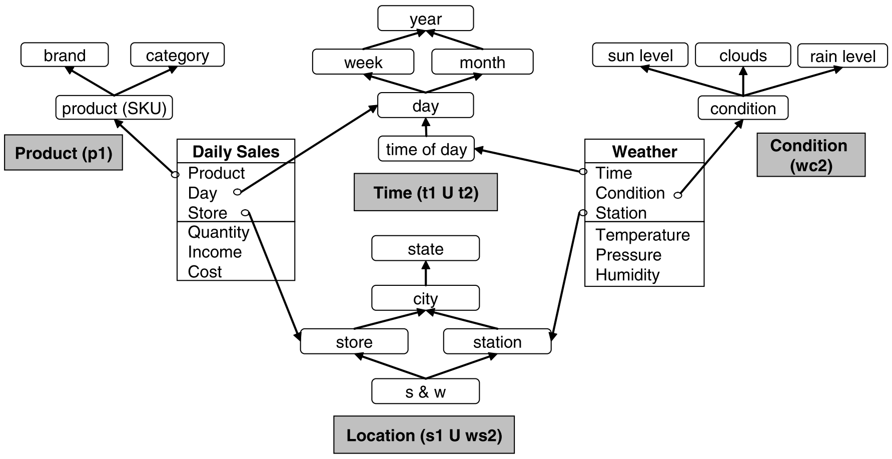


Fig. 11 The integration based on the merged dimensions

level of the time dimension and at the city level on the location dimension, and (ii) aggregate the Sales data mart at the city level. That is, we need to apply the DMA expressions reported above over the original data marts. However, differently from the other case, further combinations of the two data marts are possible. For instance, we can correlate sales and whether conditions on a weekly base.

We close this section by observing that the approach illustrated in this section can also be adopted when we wish to extend local dimensions with data from external data sources, in order to extend local querying capabilities. For instance, with the goal of performing further selections and groupings, as suggested in [25]. As an example, the Sales data mart could be integrated with an external and more sophisticated location dimension in order to select, for instance, sales in cities having more than 100,000 inhabitants.

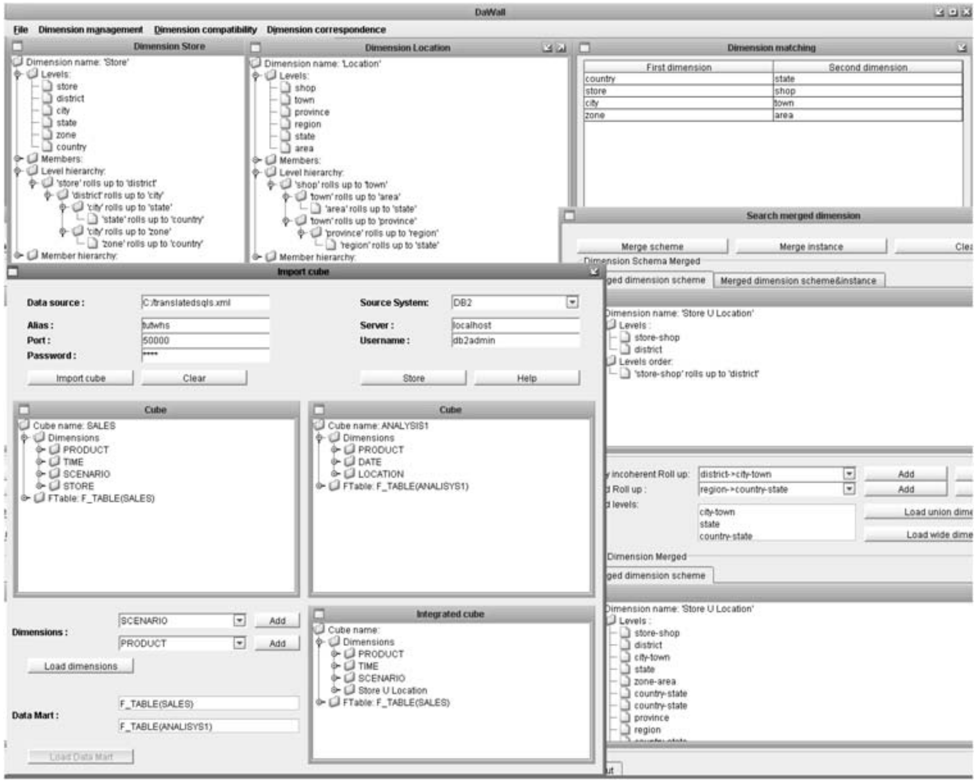
6 The integration tool

The various techniques described in the previous sections have been implemented in an interactive tool called DaWaII (Data Warehouse IntegratIon) that has been presented as a demo in [10]. Screen-shots of the current version of this tool are reported in Fig. 12. More information about the tool can be found on the Web site: `http://torlone.dia.uniroma3.it/dawaii/`.

6.1 Features of the system

DaWaII supports various activities related to the integration of multidimensional databases. More specifically, it provides the following general functionalities.

- Schemes of data marts stored in a variety of external systems (currently DB2, Oracle, and SQL Server) can be accessed and imported in DaWaII. Metadata describing cubes and dimensions in these systems are translated into a uniform internal format based on the $\mathcal{MD}$ model.



Dimension name: 'Store'
Levels:
store
district
city
state
zone
country
Members:
Level hierarchy:
'store' rolls up to 'district'
'district' rolls up to 'city'
'city' rolls up to 'state'
'state' rolls up to 'country'
'city' rolls up to 'zone'
'zone' rolls up to 'country'
Member hierarchy:

Dimension name: 'Location'
Levels:
shop
town
province
region
state
area
Members:
Level hierarchy:
'shop' rolls up to 'town'
'town' rolls up to 'area'
'area' rolls up to 'state'
'town' rolls up to 'province'
'province' rolls up to 'region'
'region' rolls up to 'state'
Member hierarchy:

First dimension	Second dimension
country	state
store	shop
city	town
zone	area

Dimension Schema Merged

Dimension name: 'Store U Location'
Levels :
store-shop
district
Levels order:
'store-shop' rolls up to 'district'

Data source : C:/translatedsqls.xml
Alias : tutwhs
Port : 50000
Password : ****
Source System: DB2
Server : localhost
Username : db2admin

Cube name: SALES
Dimensions
PRODUCT
TIME
SCENARIO
STORE
FTable: F_TABLE(SALES)

Cube name: ANALYSIS1
Dimensions
PRODUCT
DATE
LOCATION
FTable: F_TABLE(ANALISYS1)

Dimensions : SCENARIO
PRODUCT

Data Mart : F_TABLE(SALES)
F_TABLE(ANALISYS1)

Cube name:
Dimensions
PRODUCT
TIME
SCENARIO
Store U Location
FTable: F_TABLE(SALES)

incoherent Roll up: district->city-town
Roll up : region->country-state
levels: city-town
state
country-state

Dimension Merged

Dimension name: 'Store U Location'
Levels :
store-shop
district
city-town
state
zone-area
country-state
country-state
province
region

Fig. 12 The user interface of DaWaII

- Matchings between heterogeneous dimensions can be specified manually by the user by means of a graphical interface.
- Possible matchings between levels of heterogeneous dimensions can be proposed automatically by DaWaII. As it will be illustrated in more details in the following subsection, this feature relies on a number of (rather standard) heuristics that try to infer whether two levels of different dimensions can refer to the same concept.
- A dimension matching can be tested for coherence, consistency, and soundness. The checks for soundness and for consistency are not based merely on name equality, but also on name matching according to known similarity measures. Also this issue is discussed in the next subsection. If one of these tests fails, a detailed report on the corresponding cause is returned.
- The common data in two matched dimensions can be derived, according to the loosely integration approach. This function generates two new virtual dimensions that can be attached to an imported data mart.
- Two matched dimensions can be merged, according to the tightly integrated approach. This function generates a new materialized dimension that can be either attached to an imported data mart or exported to an external system.
- Drill-across queries over heterogeneous data marts can be specified. These operations make use of the new dimensions that have been built according to either the tightly coupled approach or the loosely coupled one and that are shared by the data marts involved in the query.
- Dimensions and cubes operated by DaWaII can be exported to external systems.

6.2 System architecture

The basic components of the architecture of DaWaII are reported in Fig. 13. A number of data warehouses stored in external Data Warehouse Management Systems (DWMS) are accessed by the tool through a collection of DW Interfaces (DWI). For each DWMS, the corresponding DWI is able to:

- Extract meta data describing the sources.
- Translate these descriptions into an internal representation based on the multidimensional model described in Sect. 3 and store these representations in a local data dictionary.
- Import/export dimension members to/from a local data repository.

The Dimension Manager is in charge to specify and verify matching between dimensions stored in the data dictionary. Matchings can be either specified manually by the user or proposed automatically by the system. This work is performed by a Match Maker that implements known inference techniques for the automatic derivation of mappings between levels of different dimensions. More specifically, the Match Maker module adopts both:

- A top-down approach, based on the use of a data dictionary over the level names (Wordnet [12], available at gt;lt;http://wordnet.princeton.edu), to identify relationships between level names; and
- A bottom-up approach, based on the identification of shared members between pair of levels, under the assumption that two concepts that take values from the

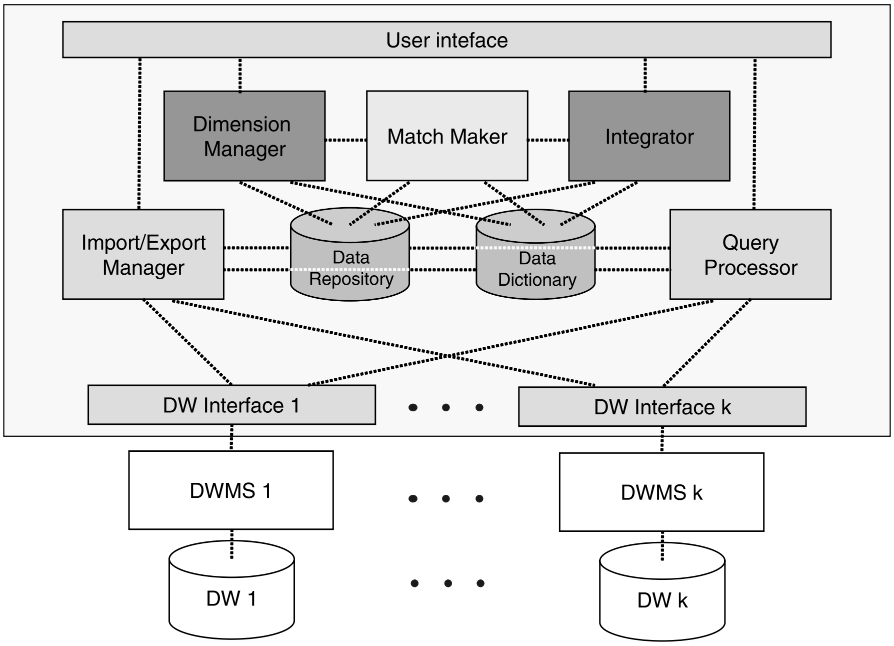


Fig. 13 The architecture of DaWaII

same domain are more likely to be similar than if they take values from completely different domains.

We stress however the fact that a deep investigation of such strategies is outside the original goal of the project.

The Integrator performs the actual integrations of pair of dimensions according to either the loosely coupled approach or the tightly coupled one using the algorithms presented in Sect. 4. In the former case, new dimension definitions are generated and stored in the data dictionary for later use. In the latter, a new dimension is built and the corresponding members are stored in the local data repository. The d-chase procedure, which is used in both approaches, takes advantage of the Match Maker to equates names of members from different dimensions.

Finally, the Query Processor receives requests of drill-across queries over autonomous data marts and, on the basis of the information available in the internal repositories, performs queries to the external systems through the corresponding DW interfaces. Actually, it should be said that the query facility has been added to the system just for testing purposes. In fact, as we have clarified in the Introduction, our goal is not development of yet another data warehouse system but rather of a CASE tool able to support the user in the design of integrated data marts, built over heterogeneous sources, that can be later queried by a suitable (and efficient) OLAP systems.

6.3 Usability and effectiveness

DaWaII has been used in different practical applications with very satisfactory results. Some notable examples are the following:

- We have used the tool for a telecommunication company to merge a number of data marts developed autonomously by different designers but using the same DBMS. In this case, it was quite easy to integrate similar dimensions presenting structural and syntactic differences. The tight integration strategy has been preferred in this application since the goal was to built a new, homogeneous data warehouse including the various multidimensional sources.
- The tool has been used in our University to combine some analyses on students done on the local data warehouse with demographical data published by ISTAT (the National Institute of Statistics), which is the main supplier of official statistical information in Italy. In this application the main problem was the construction of a DW Interface (see Sect. 6) over semi-structured files extracted by the ISTAT Web site. After this, it was however simple to combine local and global data with our system to known, for instance, the percentage of students coming from a geographical area with other Universities.
- We have been involved in a large research project in Italy with different University. During the application for funding of this project, DaWaII has been very effective to integrate several multidimensional repositories of biographical data to calculate the publications of the participants by, e.g., the year of publication and by funding support.

In almost all our use cases, the tool has demonstrated its effectiveness. The good degree of usability of the system has been confirmed by the users of the tool and by the attendees of our demo sessions [10]. With respect to the efficiency of the tool, we stress the fact that the main capabilities of DaWaII are the integration of dimensions and the construction of integrated view over local repositories of data. According to the complexity results discussed in Sect. 4, the basic operations can be always performed very efficiently, since they operate over dimensions whose size is very small with respect to the number of facts of a data warehouse. Conversely, the actual queries over the underlying fact databases are basically demanded by DaWaII to external systems. This is because, as we have said in the previous section, our goal is the development of a CASE integration tool able to support the user in the design phase of integrated data marts, which can be later queried by a efficient query engines.

7 Conclusions

In this paper we have illustrated the development of a tool for the integration of heterogeneous multidimensional databases. We have first addressed the problem from a conceptual point of view, by introducing the desirable properties of coherence, soundness and consistency that "good" matchings between dimensions should enjoy.

We have then presented two practical approaches to the problem that refer to the different scenarios of loosely and tightly coupled integration. We have shown that, if possible, both approaches guarantee the fulfillment of the above properties. To this end, we have introduced a practical tool, the chase of dimensions, that can be effectively used in both approaches to compare the content of the dimensions to integrate.

We have finally developed a tool to test on the field the effectiveness of the overall approach. Under this aspect, the tests we have done on real world scenarios are very satisfactory and the system is also very efficient [10].

In order to focus on the main aspects of the problem, we have intentionally left out the case in which data marts to be integrated present some anomaly, such as the presence of non-strict hierarchies or many-to-many relationships between facts and dimensions [29]. Adaptations and extensions of the integration techniques presented here in order to deal with such and further scenarios are subject of future research.

As a final comment, we believe that the techniques presented in this paper can be generalized to much more general contexts in which, similarly to the scenario of this study, we need to integrate heterogeneous sources and we possess a taxonomy of concepts that describe their content. As a matter of fact, we note that dimensions have structural and functional similarities with *ontologies*, which provide descriptions of concepts in a domain and are used to share knowledge. It turns out that some of the notions and the techniques presented here can provide a contribution to the problem of integrating generic information sources using ontologies. This is subject of current investigation.

Acknowledgements We would like to thank Luca Cabibbo, who has contributed to the preliminary phase of this research, and Ivan Panella, who has been involved in the development of DaWaII. We would like also to thank an anonymous reviewer for his/her helpful comments.

References

1. Abelló, A., Samos, J., Saltor, F.: On relationships offering new drill-across possibilities. In: Proc. of 5th ACM Int. Workshop on Data Warehousing and OLAP (DOLAP 2002), pp. 7–13, 2002
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison–Wesley, Reading (1995)
3. Aho, A.V., Sagiv, Y., Ullman, J.D.: Efficient optimization of a class of relational expressions. ACM Trans. Database Syst. **4**(4), 435–454 (1979)
4. Atzeni, P., Ceri, S., Paraboschi, S., Torlone, R.: Database Systems: Concepts, Languages and Architectures. McGraw–Hill, New York (1999)
5. Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database scheme integration. ACM Comput. Surv. **18**(4), 323–364 (1986)
6. Cabibbo, L., Torlone, R.: A logical approach to multidimensional databases. In: Proc. of 6th Int. Conference on Extending Database Technology (EDBT'98), pp. 183–197. Springer, Berlin (1998)
7. Cabibbo, L., Torlone, R.: From a procedural to a visual query language for OLAP. In: Proc. of 10th Int. Conference on Scientific and Statistical Database Management (SSDBM'98), pp. 74–83, 1998
8. Cabibbo, L., Torlone, R.: On the integration of autonomous data marts. In: Proc. of 16th Int. Conference on Scientific and Statistical Database Management (SSDBM'04), pp. 223–234, 2004
9. Cabibbo, L., Torlone, R.: Integrating heterogeneous multidimensional databases. In: Proc. of 17th Int. Conference on Scientific and Statistical Database Management (SSDBM'05), pp. 205–214, 2005
10. Cabibbo, L., Panella, I., Torlone, R.: DaWaII: a tool for the integration of autonomous data marts. In: Proc. of 22nd Int. Conference on Data Engineering (ICDE'06), Demo session, 2006
11. Elmagarmid, A., Rusinkiewicz, M., Sheth, A.: Management of Heterogeneous and Autonomous Database Systems. Morgan Kaufmann, San Mako (1999)
12. Fellbaum, C. (ed.): WordNet: a Lexical Database for the English Language. MIT Press, Cambridge (1998)

13. Gal, A., Anaby-Tavor, A., Trombetta, A., Montesi, D.: A framework for modeling and evaluating automatic semantic reconciliation. VLDB J. **14**(1), 50–67 (2005)
14. Honeyman, P.: Testing satisfaction of functional dependencies. J. ACM **29**(3), 668–677 (1982)
15. Hull, R.: Managing semantic heterogeneity in databases: a theoretical perspective. In: Proc. of 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems, pp. 51–61, 1997
16. Jensen, M.R., Møller, T.H., Pedersen, T.B.: Specifying OLAP Cubes on XML Data. J. Intell. Inf. Syst. **17**(2-3), 255–280 (2001)
17. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. 2nd edn. Wiley, New York (2002)
18. Lenzerini, M.: Data integration: a theoretical perspective. In: Proc. of 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems, pp. 233–246, 2002
19. Maier, D., Mendelzon, A.O., Sagiv, Y.: Testing implications of data dependencies. ACM Trans. Database Syst. **4**(4), 455–468 (1979)
20. Malvestuto, F.M.: The classification problem with semantically heterogeneous data. In: Proc. of ACM SIGMOD Int. Conference on Management of Data, pp. 157–176, 1988
21. Malvestuto, F.M., Zuffada, C.: The derivation problem for summary data. In: Proc. of 4th Int. Conference on Scientific and Statistical Database Management (SSDBM'88), pp. 82–89, 1988
22. Miller, R.J. (ed.): Special issue on integration management. IEEE Bull. Tech. Comm. Data Eng. **25**(3), (2002)
23. Miller, R.J., Hernández, M.A., Haas, L.M., Yan, L., Ho, C.T.H., Fagin, R., Popa, L.: The Clio project: managing heterogeneity. SIGMOD Rec. **30**(1), 78–83 (2001)
24. Pedersen, T.B., Shoshani, A., Gu, J., Jensen, C.S.: Extending OLAP querying to external object databases. In: Proc. of 9th Int. Conference on Information and Knowledge Management, pp. 405–413, 2000
25. Pedersen, D., Riis, K., Pedersen, T.B.: XML-Extended OLAP Querying. In: Proc. of 14th Int. Conference on Scientific and Statistical Database Management (SSDBM'02), pp. 195–206, 2002
26. Redner, R., Walker, H.: Mixture densities, maximum likelihood and the EM algorithm. SIAM Rev. **26**(2), 195–239 (1984)
27. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4), 334–350 (2001)
28. Sato, H.: Handling summary information in a database: derivability. In: Proc. of ACM SIGMOD International Conference on Management of Data, pp. 98–107, 1981
29. Torlone, R.: Conceptual models for multidimensional databases. In: M. Rafanelli (ed.) Multidimensional Databases, pp. 69–90, Idea Group Publ. (2002)
30. Torlone, R., Panella, I.: Design and development of a tool for integrating heterogeneous data warehouses. In: Proc. of 7th Int. Conference on Data Warehousing and Knowledge Discovery (DaWaK 2005), pp. 105–114, 2005
31. Yin, X., Pedersen, T.B.: Evaluating XML-extended OLAP queries based on a physical algebra. In: Proc. of 7th ACM Int. Workshop on Data Warehousing and OLAP (DOLAP'04), pp. 73–82, 2004