Integrating Heterogeneous Multidimensional Databases

Luca Cabibbo and Riccardo Torlone
Dipartimento di Informatica e Automazione
Università Roma Tre
{cabibbo, torlone}@dia.uniroma3.it

Abstract

In this paper we present a number of techniques that can be at the basis of a practical integration tool for multidimensional databases. We start by addressing 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 sources and perform drill-across queries 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 finally show how these techniques can be actually used to perform drill-across operations over heterogeneous multidimensional information sources.

1 Introduction

The problem of integrating heterogeneous multidimensional databases arises in common scenarios in which information from autonomous (i.e., independently developed and operated) data marts need to be combined. A common practice for building a data warehouse is indeed to implement a series of integrated data marts, each of which provide a dimensional view of a single business process. These data marts should be based on conformed (i.e., 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. The need for combining autonomous data marts arises in other common cases. 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) data warehouses.

In an earlier paper [5], we have introduced and investigated a fundamental notion underlying data mart integration: dimension *compatibility*. Intuitively, two dimensions (belonging to different data marts) are compatible if their common information is consistent. We have shown that dimension compatibility gives the ability to correlate, in a correct way, multiple data marts by means of *drill-across queries* [9], based on joining data over common dimensions. Building on this preliminary study, in this paper we introduce a number of notions and algorithms that can be used in a practical integration tool for multidimensional databases, similarly to how Clio [12] supports heterogeneous data transformation and integration.

We start from the problem of integrating a pair of autonomous dimensions and identify a number of desirable properties that a *matching* between dimensions (that is, a one-to-one correspondence between their levels) should satisfy: the *coherence* of the hierarchies on levels, the *soundness* of the paired levels, according to the members associated with them, and the *consistency* of the functions that relate members of different levels within the matched dimensions. We propose two different approaches to the problem of integration that try to enforce matchings satisfying the above properties. The first approach 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. This approach supports drill-across queries, to be performed over the original sources. The goal of the second approach is rather merging the sources (intuitively, making the union) and refers to a scenario of tightly coupled integration, in which we need 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.

We believe that the proposed techniques can be applied

in more general contexts in which there is the need to integrate generic heterogenous data sources and we have at our disposal taxonomies of concepts describing the sources (e.g, ontologies).

The concept of compatibility among dimensions in a data warehouse has been discussed, under the name of "conformity", by Kimball [9] in the context of data warehouse design. Our notion of compatibility is actually more suitable to autonomous multidimensional data integration than the notion of conformity since we consider an "integration" perspective rather than a "design" one. The integration of heterogenous databases has been studied in the literature extensively (see, e.g., [6, 7, 10, 13, 17]). In this paper, we take apart the general aspects of the problem and concentrate our attention on the specific problem of integrating *multidimensional* data. Differently from the general case, this problem can be tackled in a more 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. To our knowledge, the present study is the first systematic approach to this problem. A somehow related issue is the *derivability* of summary data from heterogeneous data sets in the context of statistical databases [11, 18]. Some work has been done on the problem of integrating data marts with external data, stored in various formats: XML [8, 14] and object-oriented [15]. This is related to our tightly coupled approach to integration, where dimensions are "enriched" with external data. On the other hand, our loosely coupled approach to integration is related to the problem of *drill-acrossing* [1]. Finally, the chase of dimensions can be viewed as 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 [16]. However, the goal of these studies is different from ours.

The paper is organized as follows. In Section 2 we recall a multidimensional model that will be used throughout the paper. In Section 3 we present the notion of dimension matching and provide a basic tool, called d-chase, for the management of matchings. In Section 4 we illustrate two techniques for dimension integration and, in Section 5, we describe how they can be used to integrate data marts. Finally, in Section 6, we sketch some conclusions.

2 Preliminaries

2.1 A dimensional data model

In this section, we will briefly recall the $\mathcal{MD}$ data model [4], a multidimensional conceptual data model. It generalizes the notions commonly used in multidimensional analysis or available in commercial OLAP systems and, for this reason, is adopted as a basic framework for our study. $\mathcal{MD}$ is based on two main constructs: the dimension and the data mart.

Definition 2.1 (Dimension) A *dimension* d is composed of:

- a *scheme* $S(d)$, made of: (i) a finite set $L = \{l_1, \ldots, l_n\}$ of *levels*, and (ii) a partial order $\preceq$ on L (if $l_1 \preceq l_2$ we say that l_1 *rolls up to* l_2), and
- an *instance* $I(d)$, made of: (i) a function m associating *members* with levels; and (ii) 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 $\perp$ (wrt $\preceq$) whose members represent real world entities that we call *basic*.[1] Members of 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 2.2 (Data mart) A *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 2.1 *Figure 1 shows a* Sales *data mart that represents daily sales of products in a chain of stores.*

It is worth noting that, according to the traditional database terminology, the $\mathcal{MD}$ is a *conceptual* data model and therefore its schemes can be implemented using several *logical* data models [3].

[1] In [5], we called them *ground* members.

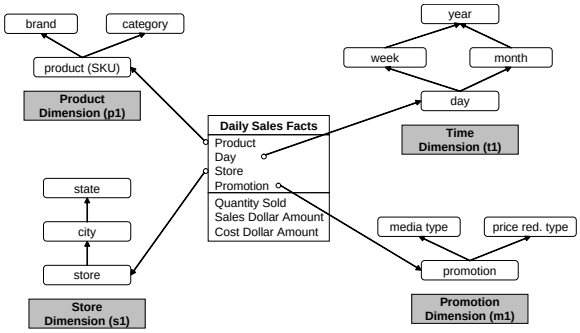

Figure 1. Sales data mart

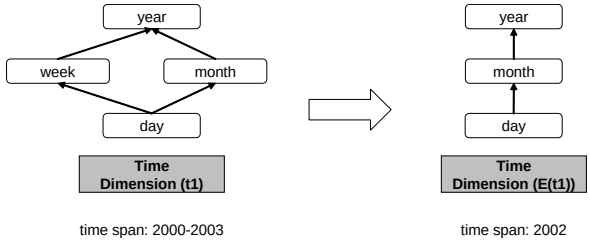

Figure 2. Application of a DA expression

2.2 An algebra for dimensions

Let d denote a dimension having scheme $(L, \preceq)$ and instance (m, ρ). The *dimension algebra* (*DA*) can be used to manipulate dimensions and is based on three operators, as follows.

Definition 2.3 (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: (i) the scheme of d' is the same of d and (ii) the instance of d' contains: the basic members in G, the members of d that can be reached from them by applying roll-up functions in ρ, the restriction of the roll-up functions of d to the members of d'.

Definition 2.4 (Projection) Let X be a subset of the levels of d including $\perp_d$. The *projection* $\pi_X(d)$ of d over X is the dimension d' such that: (i) the scheme of d' contains X and the restriction of $\preceq$ to the levels in X, (ii) 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 2.5 (Aggregation) Let l be a level in L. The *aggregation* $\psi_l(d)$ of d over l is the dimension d' such that: (i) the scheme of d' contains l, the levels of d to which l rolls up, and the restriction of $\preceq$ to these levels, and (ii) 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.

Example 2.2 *Let us consider the time dimension $t1$ of the data mart in Figure 1 and let D_{2002} denote the days that belong to year 2002. The DA expression $E = \pi_{day,month,year}(\sigma_{O_{2002}}(t1))$ generates a new dimension with level day, month and year having as basic members all the days of 2002 (see Figure 2).*

The following is a desirable property of DA expressions.

Definition 2.6 (Lossless expression) A DA expression E over a dimension d is *lossless* if for each member o in $E(d)$, all the members that roll up to o in d belong to $E(d)$.

In [5] we have shown that the satisfaction of this property prevents inconsistencies between aggregations over d and aggregations over $E(d)$.

DA expressions involving only projections and aggregations are always lossless [5]. 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 what follows, d_1 and d_2 denote two dimensions, belonging to different data marts, having scheme $S(d_i) = (L_i, \preceq_i)$ and instance $I(d_i) = (m_i, \rho_i)$, respectively.

3.1 Dimension matching and its properties

Let us start with the basic notion of dimension matching.

Definition 3.1 (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.

A number of desirable properties can be defined over a matching between dimensions.

Definition 3.2 (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 $\mu(l) \preceq_2 \mu(l')$;

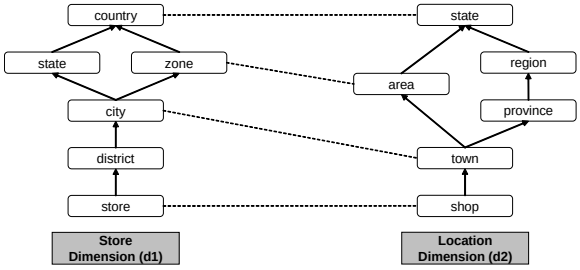

Figure 3. A matching between two dimensions

- **Soundness:** μ is *sound* if, for each level $l \in L_1$ on which μ is defined, $m_1(l) = m_2(\mu(l))$;[2]
- **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.

Note that coherence means order preservation, soundness means member set preservation, and consistency means roll-up functions preservation.

Example 3.1 *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. The matching is coherent. If the mapped levels have the same members it is also sound. Consistency follows from the equivalence of the roll-up functions between levels.*

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 3.3 (Dimension Compatibility) Two dimensions d_1 and d_2 are *compatible* if there exist two lossless DA expressions E_1 and E_2 over d_1 and d_2, respectively, such that there is a perfect matching between $E_1(d_1)$ and $E_2(d_2)$. In this case we say that d_1 and d_2 are compatible using E_1 and E_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 [5].

[2]Note that, for simplicity, we follow a *conceptual* approach, under which two levels coincides if they are populated by the same real world entities. In a *logical* approach this notion would be based on a bijection between the identifiers representing the entities.

Example 3.2 *The dimensional matching reported in Figure 3 can be made perfect by applying the following expressions to d_1 and d_2:*

$$\pi_{store,city,zone,country}(\sigma_{m_1(\text{store}) \cap m_2(\text{shop})}(d_1)),$$

$$\pi_{shop,town,area,state}(\sigma_{m_1(\text{store}) \cap m_2(\text{shop})}(d_2)).$$

provided that the roll-up functions of the two dimensions are consistent. If the original dimensions had basic members in common and the selection over them made the two expressions lossless, then they would be compatible.

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 $L = l_1, \ldots, l_k$ 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 3.4 (Matching Tableau (MT)) The *matching tableau* over d_1, d_2 and μ, denoted by $T_\mu[d_1, d_2]$, is a tableau over $L = L_1 \bigcup \mu(L_2)$ 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 3.3 *A possible matching tableau for the matching between dimensions in Figure 3 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	E1	France	Marais	v_7	v_8	v_9
1mo	Rome	E1	Italy	v_{10}	v_{11}	RM	Lazio
1st	NewYork	U1	USA	v_{12}	v_{13}	v_{14}	v_{15}
1er	Paris	E1	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).*

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 3.5 (D-chase) The d-chase of a tableau T, denoted by $DCHASE_\rho(T)$, is a tableau obtained from T and a set of roll-up functions ρ by applying all valid chase steps exhaustively to T.

Example 3.4 *By applying the d-chase procedure to the matching tableau of Example 3.3 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	E1	France	Marais	v_7	75	IledeFr
1mo	Rome	E1	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 follows from general properties of d-chase, is the following.

Lemma 3.1 *The d-chase process terminates on any input with a unique end result.*

The following result states that the d-chase provides an effective way to test for consistency.

Theorem 3.5 *A matching μ between two dimensions d_1 and d_2 is consistent if and only if $DCHASE_{\rho_1 \cup \mu(\rho_2)}(T_\mu[d_1, d_1]) \neq T_\infty$.*

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 3.6 (Total projection) The *total projection* of T over S, denoted by $\pi^{\downarrow}_S(T)$, 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.

Let d be a dimension and μ a matching between d and any other dimension d'. We can easily show the following.

Lemma 3.2 $I(d) \subseteq \pi^{\downarrow}_{S(d)}(DCHASE_{\rho \cup \mu(\rho')}(T_\mu[d, d']))$.

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. As noted above, this situation occurs when, in a tuple for a member in d, the d-chase promotes a variable to a member of 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.

4.1 A loosely coupled approach

In a *loosely coupled integration* scenario, we need to identify the common information between various data sources and perform drill-across queries over the original sources. Therefore, our goal is just 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 Figure 4.

First of all, the algorithm selects the levels L of d_1 involved in the matching μ (Step 1). Then, for each minimal

Algorithm 1
Input: two dimensions d_1 and d_2 and a matching μ;
Output: two expressions E_1 and E_2 that make μ perfect;
begin
1) $L :=$ the levels of d_1 involved in μ;
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) $M := m_1(l_m) \bigcap m_2(\mu(l_m))$;
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** $\exists\, m' \in m_{1,2}(l')$ **such that** $l' \preceq_{1,2} l$ **and** $\rho^{l' \to l}_{1,2}(m') = m$ **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

Figure 4. An algorithm for deriving the common information between two dimensions.

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 testing for coherence of the matching according to Definition 3.2. 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 ($\mu(L)$). Since no selection is involved, by a result in [5], these expressions are lossless.

The rest of the algorithm aims at finding the selection of members that, applied to E_1 and E_2, identifies common data 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). According to Theorem 3.5, this corresponds to a test of consistency for the restriction of μ to the levels in L.

As we have noticed at the end of Section 3, when the d-chase promotes a variable to a member, this means that a previously unknown value in one dimension has been identified in the other dimension. To preserve soundness, this event asks for the addition of this member in the original dimension: this is implemented by Step 10.

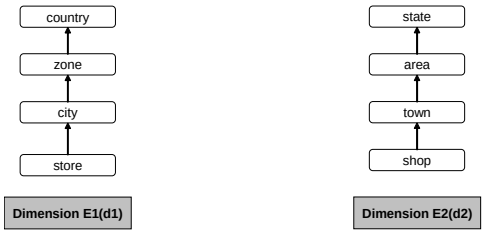

Figure 5. The dimensions generated by Algorithm 1 on the matching in Figure 3

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 2.6). 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 4.1 *Let us consider the application of Algorithm 1 to the dimensions and the matching in Figure 3, assuming that that the dimensions are populated by the members of Example 3.3. 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	E1	France	Marais	v_7	75	IledeFr

This tableau contains the member E1 at the zone level to which a member of d_2 *rolls up (the store* 1mo*), but is not present in the tableau. It follows that in Step 12 the second row is deleted and we obtain as output of the algorithm the following final expressions:*

$$\sigma_{\{1st\}}(\pi_{store,city,region,country}(d_1)),$$

$$\sigma_{\{1st\}}(\pi_{shop,town,area,state}(d_2)).$$

The schemes of the dimensions we obtain by applying these expressions to the original dimensions are reported in Figure 5.

By construction, and according to the results of the previous section, we can state the following.

Theorem 4.2 *The execution of Algorithm 1 over two dimensions* d_1 *and* d_2 *and a matching* μ *between them returns*

Algorithm 2
Input: two dimensions d_1 and d_2 and a matching μ;
Output: a new dimension d that embeds d_1 and d_2;
begin
1) $L :=$ the levels of d_1 involved in μ;
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) $L := L_1 \bigcup \mu(L_2)$;
4) $\preceq := (\preceq_1 \bigcup \mu(\preceq_2))^+$;
5) **if** $\preceq$ has several minimal levels
then
6) $d'_1 :=$ d_1 augmented with a new bottom level $\perp'_1$;
7) $d'_2 :=$ d_2 augmented with a new bottom level $\perp'_2$;
8) $\mu' := \mu \bigcup \{(\perp'_1, \perp'_2)\}$;
9) $L := L \bigcup \perp'_1$;
10) $\preceq := (\preceq'_1 \bigcup \mu(\preceq'_2))^+$;
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

Figure 6. An algorithm for merging two dimensions.

two expressions E_1 and E_2 if and only if d_1 and d_2 are compatible using E_1 and E_2.

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 *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 Figure 6.

First of all, similarly to Algorithm 1, Algorithm 2 performs a check for coherence of the input matching. If the test is successful, it then 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. 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.

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 (clearly, this can be done without actually modifying the original dimensions). These levels have as members copies of the basic members of $\perp_1$ and $\perp_2$, suitably renamed so that the intersection of the two sets of copies is empty. Then, two new roll-up functions $\rho^{\perp'_1 \to \perp_1}$ and $\rho^{\perp'_2 \to \perp_2}$ mapping each copy to the corresponding member are added to the instances of the dimensions. Finally, 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). All of this guarantees the uniqueness of the bottom level for L without generating undesirable inconsistencies.

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

Example 4.3 *Let us consider again the matching between dimensions in Figure 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 Figure 7. If the dimensions are populated by the member of Example 3.3, the output instance contains all the members occurring in the chased matching tableau reported in Example 3.4.*

We say that a dimension d *embeds* another dimension d' if there exists a DA expression E such that $E(d) = d'$. By construction and on the basis of the discussion above, we can state the following result.

Theorem 4.4 *The execution of Algorithm 2 over two dimensions d_1 and d_2 and a matching μ between them returns a new dimension d embedding 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

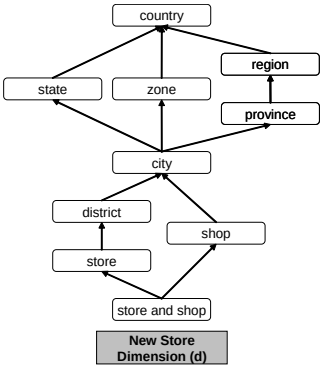

Figure 7. The dimension generated by Algorithm 2 on a variant of the matching in Figure 3

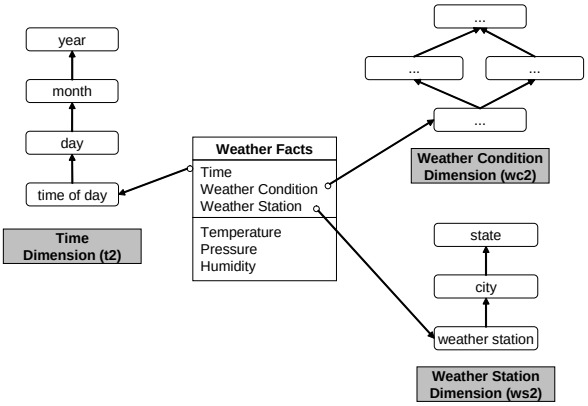

Figure 8. Weather data mart

of the dimensions involved. Hence, we can make for this algorithm the same considerations done for Algorithm 1.

5 Data mart integration

In this section we discuss, by means of some examples, how the techniques described in Section 4 can be used in data warehouse 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 [9]. These queries are based on joining different data marts over common dimensions [9]. 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.

The loosely coupled approach supports drill-across queries between data marts, in that it aims at identifying the intersection between their dimensions. Actually, the proposed algorithm also checks for the *quality* of such intersection; in particular, dimension compatibility (e.g., the

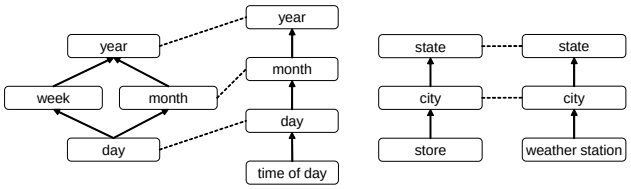

Figure 9. A matching between time and location dimensions

existence of a "perfect" intersection). As discussed in [5], this is a necessary condition for obtaining meaningful results when aggregations must be computed over data marts.

Assume, for instance, that we wish to integrate the Sales data mart reported in Figure 1 with the data mart storing weather information reported in Figure 8, in order to correlate sales of products with weather conditions. The integration between these data sources can be based on the matchings between the time ($t1$ and $t2$) and the location dimensions ($s1$ and $ws2$) as indicated in Figure 9.

The application of Algorithm 1 to this input checks for compatibility of dimensions and returns the following pairs of expressions. The first two expressions select the members in common in the time dimensions:

$$\pi_{day,month,year}(\sigma_{\mathrm{day}_{t1} \cap \mathrm{day}_{t2}}(t1)),$$
$$\psi_{day}(\sigma_{\mathrm{day}_{t1} \cap \mathrm{day}_{t2}}(t2)).$$

where $\mathrm{day}_{t1} \cap \mathrm{day}_{t2}$ denotes the days in common that make the matching between $t1$ and $t2$ perfect. The other pair of expressions select the members in common in the location dimensions:

$$\psi_{city}(\sigma_{\mathrm{city}_{s1} \cap \mathrm{city}_{ws2}}(s1)).$$
$$\psi_{city}(\sigma_{\mathrm{city}_{s1} \cap \mathrm{city}_{\mathrm{ws2}}}(ws2)).$$

where $\mathrm{city}_{s1} \cap \mathrm{city}_{ws2}$ denotes the cities in common that make the matching between $s1$ and $ws2$ 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. It follows that drill-across queries can be defined over the virtual view shown in Figure 10.

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 above. If we apply Algorithm 2 over the time and location dimensions and the

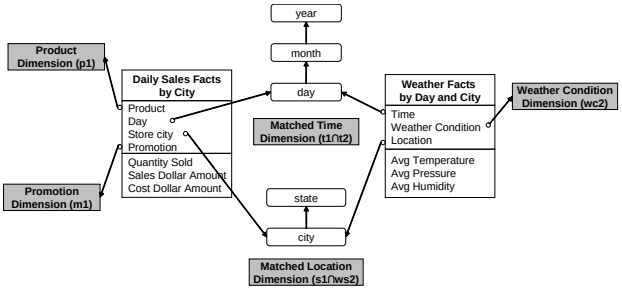

Figure 10. A virtual view over the common portion of the dimensions

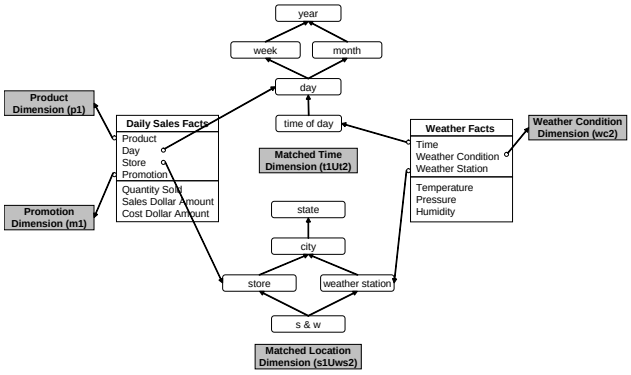

Figure 11. A materialized view over the merged dimensions

matchings in Figure 9, we generate two new dimensions that can be materialized and used for both data marts. We can then refer to the homogenous scheme reported in Figure 11 to perform drill-across queries.

Another application of the second approach is when we wish to extend local dimensions with data from external dimensions, but ignoring remote factual data, to extend local querying capabilities. For instance, to specify further selections and groupings (as suggested in [14]).

For example, consider again the Sales data mart. It could be integrated with an external and more sophisticated location dimension to select, for instance, sales in cities having more than 100.000 inhabitants.

6 Conclusion

We have proposed in this paper a number of concepts and techniques 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.

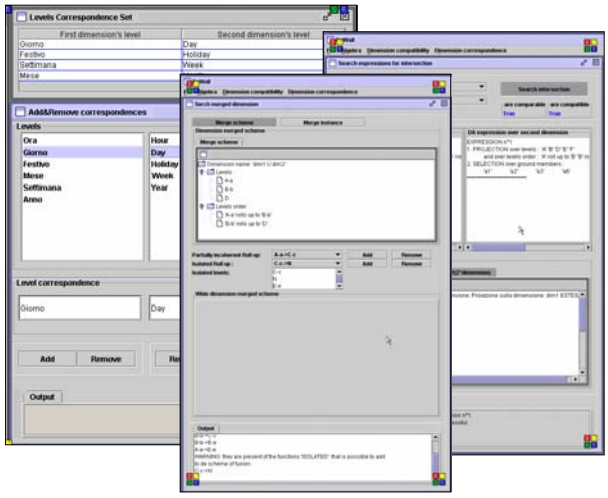

Figure 12. A prototype of the system

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.

To test our approach, we have designed and developed the first release of an interactive tool for the integration of multidimensional databases, called DaWaII (for Data Warehouse IntegratIon), that implements the proposed techniques. Specifically, this tool is able to: (i) access data marts stored in a variety of systems (DB2, Oracle, SQL Server, among others); (ii) extract from these systems metadata describing cubes and dimensions and translate these descriptions in $\mathcal{MD}$ format; (iii) specify by means of a graphical interface matchings between autonomous dimensions; (iv) test for coherence, consistency, and soundness of matchings; (v) generate the intersection between two dimensions, according to the loosely integration approach; and (vi) merge two dimensions, according to the the tightly integration approach. An hint of the graphical interface provided by this tool is reported in Figure 12.

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 heterogenous 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 integrat-

ing generic information sources using ontologies. This is subject of current investigation.

Acknowledgements

We would like to thank Ivan Panella, who is actively involved in the development of DaWaII. Moreover, we would like to thank the anonymous referees for their helpful comments and suggestions.

References

[1] A. Abelló, J. Samos, and F. Saltor. On relationships Offering New Drill-across Possibilities. In *ACM Fifth Int. Workshop on Data Warehousing and OLAP (DOLAP 2002)*, pages 7–13, 2002.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] P. Atzeni, S. Ceri, S. Paraboschi, and R. Torlone. *Database Systems: concepts, languages and architectures*. McGraw-Hill, 1999.

[4] L. Cabibbo and R. Torlone. A logical approach to multidimensional databases. In *Sixth Int. Conference on Extending Database Technology (EDBT'98), Springer-Verlag*, pages 183–197, 1998.

[5] L. Cabibbo and R. Torlone. On the Integration of Autonomous Data Marts. In *16th Int. Conference on Scientific and Statistical Database Management (SSDBM'04)*, pages 223–234, 2004.

[6] A. Elmagarmid, M. Rusinkiewicz, and A. Sheth. *Management of Heterogeneous and Autonomous Database Systems*. Morgan Kaufmann, 1999.

[7] R. Hull. Managing Semantic Heterogeneity in Databases: A Theoretical Perspective. In *16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems*, pages 51–61, 1997.

[8] M.R. Jensen, T.H. Møller, and T.B. Pedersen. Specifying OLAP Cubes on XML Data. *J. Intell. Inf. Syst.*, 17(2-3): 255–280, 2001.

[9] R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons, Second edition, 2002.

[10] M. Lenzerini. Data Integration: A Theoretical Perspective. In *21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems*, pages 233-246, 2002.

[11] F. M. Malvestuto. The Derivation Problem for Summary Data. *ACM SIGMOD International Conference on Management of Data*, pages 82–89, 1988.

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

[13] R.J. Miller (editor). Special Issue on Integration Management. *IEEE Bulletin of the Technical Committee on Data Engineering*, 25(3), 2002.

[14] D. Pedersen, K. Riis, and T.B. Pedersen. XML-Extended OLAP Querying. In *Int. Conference on Scientific and Statistical Database Management (SSDBM'02)*, pages 195–206, 2002.

[15] T.B. Pedersen, A. Shoshani, J. Gu, and C.S. Jensen. Extending OLAP Querying to External Object Databases. In *Int. Conference on Information and Knowledge Management*, pages 405–413, 2000.

[16] R. Redner and H. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.

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

[18] H. Sato. Handling Summary Information in a Database: Derivability. *ACM SIGMOD International Conference on Management of Data*, pages 98–107, 1981.