Dimension Compatibility for Data Mart Integration

Luca Cabibbo and Riccardo Torlone

Dipartimento di Informatica e Automazione
Università degli studi Roma Tre
{cabibbo,torlone}@dia.uniroma3.it

Abstract. The problem of integrating autonomous data marts arises when, e.g., a large organization (or a federation thereof) needs to combine independently developed data warehouses. It turns out that this problem can be tackled in a systematic way because of two main reasons. First, data marts are usually structured in a rather uniform way, along dimensions and facts. Second, data quality in data marts is usually higher than in generic databases, since they are obtained by reconciling several data sources. Our scenario of reference is a federation of various data marts that we need to query in a unified way by means of drill-across operations. We propose a novel notion of dimension compatibility and characterize its general properties. We then show the significance of dimension compatibility in performing drill-across queries over autonomous data marts.

1 Introduction

Nowadays, many data warehouses are developed using the dimensional modeling approach, that is, as series of coherent data marts [6]. Each data mart provides a dimensional view of a single business process and an integrated data warehouse can be built from them by using a bus architecture based on conformed (i.e., common) dimensions and facts. In large companies, however, very often different departments develop their data marts independently, and it turns out that their combination is a difficult task. Actually, the need of integrating independently developed and operated data marts arises in other common cases, for instance, when companies merge or get involved in a federated project. Another common scenario requiring the integration of autonomous data marts occurs when there is the need to combine a proprietary data warehouse with multidimensional data available on the Web [11].

Differently from the general problem of database integration [4], we believe that the integration of data marts can be tackled in a more systematic way because of two important reasons. First, any data mart is structured in a rather uniform way, along the widely accepted notions of dimension and fact. Second, data quality in a data mart is usually higher than in a generic database, since it is obtained by cleaning and reconciling several data sources. These observations suggest that the problem of integrating autonomous data marts can be mainly focused on the integration of independently developed facts and dimensions.

According to this view, in this paper we introduce and investigate a fundamental notion underlying data mart integration: *dimension compatibility.* Intuitively, two dimensions of different data marts are compatible when their common information is consistent. Similarly, two facts are compatible when their contents can be combined in a meaningful way. Having compatible dimensions and facts is important because it gives the ability to look consistently at data across data marts and to combine and correlate such data, e.g., to perform value chain analyses. In particular, *drill-across queries* are based on joining multiple data marts over common dimensions [6].

Assume for instance to have three data marts, describing promotions of products, sales of products, and store inventory levels of products, respectively. A drill-across query over all these three data marts is needed if we want to identify products that, although in promotion and available in the stores, have sold under expectations. For a drill across query, compatibility of dimensions and facts is required to obtain meaningful results. We show that drill across operations over incompatible dimensions and facts are not possible or, worse, produce invalid results. To this end, we introduce a dimension algebra that allows us to select the relevant portion of a dimension for integration purposes.

This paper provides the foundations for our ultimate goal: the development of a system for supporting the complex tasks related to the integration of autonomous data marts, similarly to the way in which a system like Clio [9] supports heterogeneous data transformation and integration.

The integration of heterogenous databases has been extensively studied in the literature (surveys on the many facets of this issue can be found in [4, 5, 7, 10, 13]). In this paper we take apart the general aspects of the problem and concentrate our attention on *multidimensional* data integration. This subject has been studied by Kimball [6] in the context of data warehouse design. In his book, he has identified the problem and has introduced, in an informal way, the notions of dimension and fact *conformity.* Our notion of compatibility has been inspired by this work, but extends and formalizes Kimball's notion of conformity in a way that, we believe, is more suitable to autonomous data mart integration. Abellò et al. [1] have investigated properties that are relevant to drill-across navigation. Although our notion of compatibility is related to such properties, the goals of the papers differ, since we refer to drill-across *queries* whereas [1] refers a weaker form of drill acrossing, specific to *interactive navigation.* Some work has been done on the problem of integrating data marts with external data stored in various formats: object-oriented [12] and XML [11]. This is clearly related to but different from our goal, since no attempt is made in these papers to combine multiple multidimensional databases.

The rest of the paper is organized as follows. In Section 2 we recall $\mathcal{MD}$, a conceptual model for multidimensional data, introduced in [2], that will be used throughout this paper. In Section 3 we present an algebra over dimensions, a tool that will be used, in Section 4, to introduce the notion of dimension compatibility. In Section 5 we investigate the relationship between compatibility and the operation of drill across between data marts. Finally, in Section 6, we sketch some conclusive remark and discuss future directions of research.

2 A dimensional data model

In this section, we describe the $\mathcal{MD}$ data model [2], a multidimensional conceptual data model that will be used throughout this paper. This choice is motivated by the fact that $\mathcal{MD}$ includes a number of concepts that generalize the notions commonly used in multidimensional analysis or available in commercial OLAP systems, e.g., dimensions, fact tables [6] and cubes [8]. Because of this, our approach can be considered rather general and can be applied to other multidimensional data models [14].

$\mathcal{MD}$ is based on two main constructs: the dimension and the f-table. A *dimension* represents a domain of real-world entities called *members*. Members of a dimension can be the days in a time interval, the products sold by a company, or a collection of stores selling these products. Each dimension is organized into a hierarchy of *levels*, corresponding to data domains grouping dimension members at different granularity. For example, products can be grouped into brands and categories, and days can be grouped into months and years. Within a dimension, members at different levels are related through a family of *roll-up functions*. A roll-up function relates the members of a pair of levels by mapping each member having a finer grain (e.g., a product) to a member having a coarser grain (e.g., a brand). An *f-table* is the conceptual counterpart of a fact table and associate *measures* to members of dimensions and are used to represent factual data. For example, the daily sales of a chain of stores can be represented by an f-table that associates with a product, a day, and a store, the number of items sold of that product, day, and store, together with the corresponding gross income and cost. In the following, we provide a more systematic presentation of these notions.

Definition 1 (Dimension). *A* dimension d *is composed of:*

- *a* scheme*, made of a finite set* $L(d) = \{l_1, \ldots, l_n\}$ *of* levels *and a partial order* $\preceq_d$ *on* $L(d)$*; if* $l_1 \preceq_d l_2$ *we say that* l_1 rolls up to l_2*;*
- *an* instance*, made of a function* m *associating a set of* members *with each level 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 $L(d)$ contains a bottom element $\perp_d$ (wrt $\preceq_d$). We shall simply write $\preceq$ instead of $\preceq_d$ whenever d is clear from the context.

Actually, each member of the bottom level $\perp_d$ of a dimension d (the finest grain for the dimension) represents a real world entity that we call *ground*. Members of other levels represent groups of ground members. For example, within a dimension of products, a ground member is a single product, whereas a member of the level *brand* describes a single brand, that is, the group composed by all the products of that brand. The *active domain* of a dimension d is the set of all the members that actually belong to the various levels of d.

For each pair of levels l_1, l_2 of a dimension d such that $l_1 \preceq_d l_2$, we assume that the following holds in any instance:

- if $o_1 \in m_d(l_1)$, then there is an $o_2 \in m_d(l_2)$ such that $\rho^{l_1 \to l_2}(o_1) = o_2$;
- if $o_2 \in m_d(l_2)$, then there is at least an $o_1 \in m_d(l_1)$ such that $\rho^{l_1 \to l_2}(o_1) = o_2$.

Therefore, for each ground member of the dimension there is a member in each aggregation level to which it rolls up. Furthermore, for each member of a non-bottom level, there is at least a ground member that rolls up to it.

Let $\{\tau_1, \ldots, \tau_k\}$ be a predefined set of *base types*, (including integers, real numbers, etc.).

Definition 2 (F-table). *An* f-table 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*, that is, a function mapping each attribute name* A_i *to a member of* l_i*; and a* fact *is a tuple over the measures of* f*, that is, a function mapping each measure name* M_j *to a value in the domain of type* τ_j. □

A collection of f-tables over the same dimensions compose a data mart.

Definition 3 (Data mart). *A* data mart *is composed of a set* D *of dimensions, and a set* F *of f-tables over the dimensions in* D. □

Example 1. Figures 1, 2, and 3 show three (autonomous) data marts, each consisting of a single f-table. This example is inspired by case studies discussed in [6]. The *Sales* data mart (Figure 1) represents daily sales of products in a chain of stores. The *Store Inventory* data mart (Figure 2) represents inventory snapshots for the same products and stores of the *Sales* data mart given on a weekly base. Finally, the *Warehouse Inventory* data mart (Figure 3) represents daily inventory snapshots for the warehouses supplying only the food and beverage products to the same stores of the *Sales* and *Store Inventory* data marts.

Note that the granularity of data in the three data marts is different. This is apparent for time data, given daily in two data marts, but weekly in the other one. Similarly, products are described as SKUs (stock keeping units, that is, products that can be sold at retail, like a can of coke) at the stores, but as packages (boxes of SKUs, e.g., a package of 20 cans of coke) at the warehouses.

Another difference is that the first two data marts contain data about many products along years 2000-2003, whereas the warehouse data mart contains only data about food and beverage products along years 2002-2003. □

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 model. For example, as described in [2], an $\mathcal{MD}$ data mart can be easily implemented as a set of star schemes [6] having a dimension table for each dimension and a fact table for each f-table.

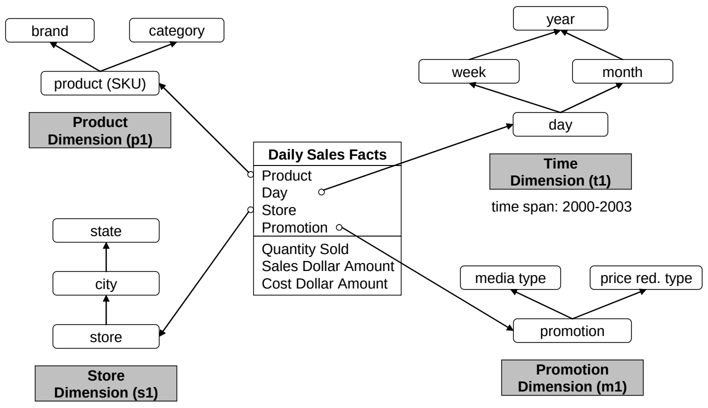

Fig. 1. Sales data mart

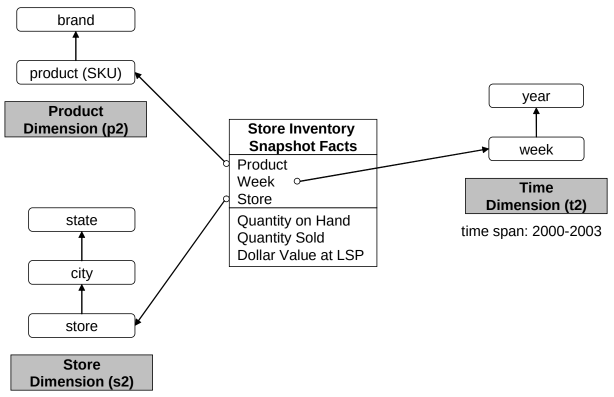

Fig. 2. Store Inventory data mart

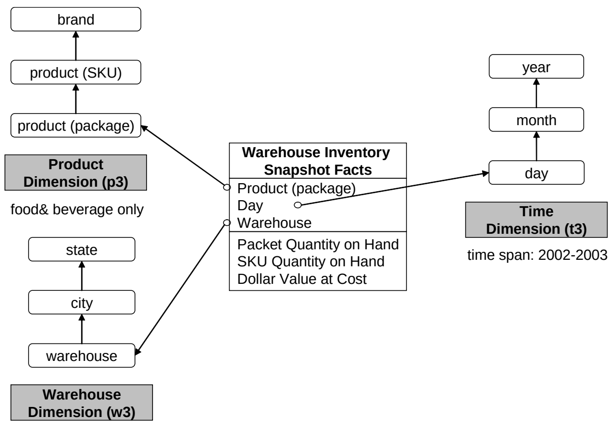

Fig. 3. Warehouse Inventory data mart

3 An algebra for dimensions

We now introduce the *dimension algebra* (*DA*), a simple algebra over dimensions that will be used to extract sub-dimensions from a given dimensions. DA is based on three operators: (i) selection, which restricts a dimension to a subset of its ground members; (ii) projection, which prunes levels and roll-up functions from a dimension; (iii) aggregation, which aggregates over a level in a dimension.

In what follows, d denotes a dimension having scheme $(L(d), \preceq)$ and instance (m, ρ).

Definition 4 (Selection). *Let S be a subset of the ground members of d. The* selection $\sigma_S(d)$ *of d over S is the dimension d' such that: (i) the scheme of d' is the same of d and (ii) the instance of d' contains: the ground members in S, the members of d that can be reached from them by applying roll-up functions in ρ, all the roll-up functions of d restricted to the members of d'.* □

Definition 5 (Projection). *Let X be a subset of the scheme of d such that: (i) $\bot_d \in X$ and (ii) if X contains $l_1 \preceq l_2$ then both l_1 and l_2 are in X. The* projection $\pi_X(d)$ *of d over X is the dimension d' such that: (i) the scheme of d' is X and (ii) the instance of d' contains: only the members of d that belong to levels in X and only the roll-up functions $\rho^{l_1 \to l_2}$ of d such that $l_1 \preceq l_2$ belong to X.* □

Definition 6 (Aggregation). *Let l be a level in $L(d)$. The* aggregation $\psi_l(d)$ *of d over l is the dimension d' such that: (i) the scheme of d' contains l, all the levels of d to which l rolls up, and the restriction of $\preceq$ to these levels, and (ii) the instance of d' contains: only the members of d that belong to levels in d' and only the roll-up functions $\rho^{l_1 \to l_2}$ of d such that $l_1 \preceq l_2$ belong to d'.* □

The simplest DA expression consists just of a dimension name, and its result is the dimension itself. More complex DA expressions can be written by applying and combining the three DA operators above to a dimension. We denote by $E(d)$ the dimension obtained by applying a DA expression E to a dimension d.

Example 2. In Example 1, the *Time* dimension t_2 of the *Store Inventory* data mart can be computed by applying an aggregation to the *Time* dimension t_1 of the *Sales* data mart, as $\psi_{week}(t_1)$ (see Figure 4). Similarly, the *Time* dimension t_3 of the *Warehouse Inventory* data mart can be computed by applying a selection followed by a projection to the *Time* dimension t_1 of the *Sales* data mart, as follows (see Figure 5):

$$\pi_{day,month,year,day \preceq month,month \preceq year}(\sigma_{O_{2002-2003}}(t_1)),$$

where $O_{2002-2003}$ denotes the days that belong to years 2002-2003. □

It is worth noting that DA expressions allow to "reduce" the scheme and/or the instance of a dimension. Indeed, the goal of a DA expression is to compute

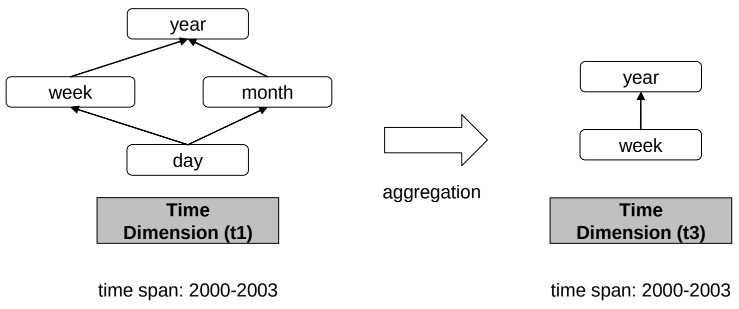

Fig. 4. Application of a DA expression

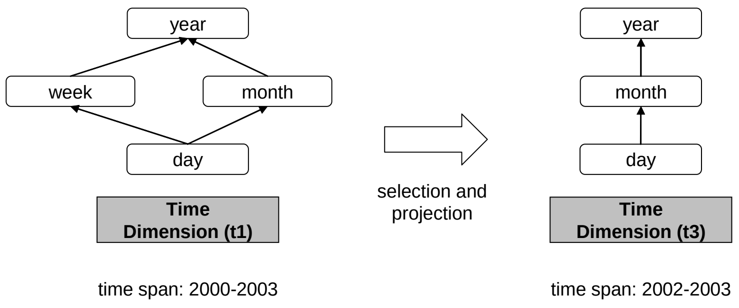

Fig. 5. Application of another DA expression

a subset of a dimension. Note also that projection and aggregation have different goals, since projections always keep the bottom level whereas non trivial aggregations drop it.

We now introduce a desirable property of DA expressions.

Definition 7 (Lossless expression). *A DA expression E over a dimension d is* lossless *if, for each each pair of ground members $o_1, o_2 \in m_d(\bot_d)$ and for each level $l \in L(d)$ such that o_1 and o_2 roll up to a same member $o \in m_d(l)$ (that is, $\rho_d^{\bot_d \to l}(o_1) = \rho_d^{\bot_d \to l}(o_2) = o$), then neither or both o_1, o_2 belong to the active domain of $E(d)$.* □

In other words, E is lossless if, whenever a member o belongs to $E(d)$, then all the members that roll up to o in d belong to $E(d)$ as well. This property is important because, if E is lossless, then aggregating an f-table over $E(d)$ yields as result a subset of the facts that can be obtained by aggregating over d, with the same measures. Otherwise, the result of aggregating over $E(d)$ could produce different results than aggregating directly over d.

It is possible to show that DA expressions involving only projections and aggregations are always lossless. 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.

4 Dimension compatibility

In this section we present our notion of compatibility among dimensions. Intuitively, two dimensions are compatible if their share some information and this common information is consistent. This is a very important requirement in drill across queries, where data marts are joined over related dimensions.

The notion of compatibility between dimensions will be introduced gradually, by first defining the stronger notion of equivalence. In what follows, d_1 and d_2 denote two dimensions, belonging to different data marts, having scheme $(L(d_i), \preceq_{d_i})$ and instance (m_i, ρ_i), respectively. Moreover, l_1 and l_2 denote two levels, $l_1 \in L(d_1)$ and $l_2 \in L(d_2)$.

Definition 8 (Level equivalence). *Two levels l_1 and l_2 are* equivalent *(written $l_1 \equiv l_2$) if they have the same members, that is, $m_1(l_1) = m_2(l_2)$.* □

Definition 9 (Dimension equivalence). *Two dimensions d_1 and d_2 are* equivalent *(written $d_1 \equiv d_2$) if there exists a bijection ϕ between $L(d_1)$ and $L(d_2)$ such that:*

- *for each level $l \in L(d_1)$, l is equivalent to $\phi(l)$;*
- *for each pair of levels $l, l' \in L(d_1)$, $l \preceq_{d_1} l'$ if and only if $\phi(l) \preceq_{d_2} \phi(l')$; and*
- *for each pair of levels $l, l' \in L(d_1)$ such that $l \preceq_{d_1} l'$, the roll-up functions $\rho_1^{l \to l'}$ and $\rho_2^{\phi(l) \to \phi(l')}$ are equal.* □

According to this definition, two equivalent dimensions represent exactly the same information, apart from differences in the choice of the names given to levels.

It is still possible that two non-equivalent dimensions have some information in common. The first requirement is the existence of an operational way to compare *portions* of dimensions. This comment leads to the following definitions.

Definition 10 (Dimension comparability). *Two dimensions d_1 and d_2 are* comparable *if there exist DA expressions E_1 and E_2 over d_1 and d_2, respectively, such that $E_1(d_1)$ and $E_2(d_2)$ are not empty and equivalent. In this case, we say that d_1 and d_2 are* comparable *using E_1 and E_2.* □

Definition 11 (Dimension intersection). *If two dimensions d_1 and d_2 are comparable using E_1 and E_2, then the dimension $E_1(d_1) \equiv E_2(d_2)$ is called an* intersection *of d_1 and d_2.* □

We are now ready to introduce our notion of compatibility between dimensions.

Definition 12 (Dimension compatibility). *Two dimensions d_1 and d_2 are* compatible *if they are comparable using two lossless DA expressions E_1 and E_2.* □

In sum, the rationale under the definition of compatibility is that: (i) the intersection of two dimensions represents their common information; (ii) DA expressions are used to compute this intersection; and (iii) lossless expressions avoid inconsistency, in a sense that will be clarified in the following section.

Example 3. Consider again the data marts of Example 1. The *Store* dimensions s_1 and s_2 of the *Sales* and *Store Inventory* data marts are equivalent. Their *Product* dimensions p_1 and p_2 are not equivalent but compatible; in fact, p_2 can be computed from p_1 as $\pi_{product(SKU),brand,product(SKU)\preceq brand}(p_1)$. The *Time* dimensions t_1 and t_2 are also compatible: t_2 can be computed as $\psi_{week}(t_1)$. The *Store* dimension s_1 and the *Warehouse* dimension w_3 are compatible, since $\psi_{city}(s_1)$ is equivalent to $\psi_{city}(w_3)$. The *Product* dimensions p_1 and p_3 are compatible too: their common part can be computed by aggregating p_3 over level *product (SKU)*, and by applying to p_1 a projection (over levels *product (SKU)* and *brand* and the roll-up relationship between them) and a selection (over food and beverage products). □

5 Drill across queries and dimension compatibility

In this section we investigate the impact of dimension compatibility on drill across queries.

We first define a drill across operator. In [3] we have defined a natural join operation $f_1 \bowtie f_2$ of two f-tables over a set of common attributes (defined on the same dimensions) whose result is the f-table having as entries the natural join (in the relational sense) of the entries of f_1 and f_2 and as facts (i.e., measures) the juxtaposition of their facts. A *drill across* operation between two f-tables f_1 and f_2 can be defined as an extension of the natural join, in which common attributes refer to different but compatible dimensions. In this case, before joining f_1 and f_2, for each pair of common attributes over compatible dimensions d_1 and d_2, we identify an intersection $d_{1\cap 2}$ of d_1 and d_2 and then aggregate f_1 and f_2 over the bottom level of $d_{1\cap 2}$.

Example 4. Consider again the data marts in Example 1 and a drill across operation over the *Sales* and *Store Inventory* data marts. They can be combined over the compatible dimensions *Product*, *Time*, and *Store*. It follows that the drill across requires an aggregation of the *Sales* data mart over the *Time* dimension at the *week* level.

Another possible drill across query is on the *Sales* and *Warehouse Inventory* data marts. Before joining them, they need to be aggregated over the common *city* level in the compatible dimensions *Store* and *Warehouse.* □

A number of anomalies can arise in the computation of a drill across query:

– some detail of the original data marts can be lost in the aggregations preceding the join, when f-tables store facts at different levels of aggregation;
– some data of the original data marts can be lost in the join, when f-tables refer to members that do not belong to the intersection of compatible dimensions.

As an example of the former type of anomaly, the drill across query over the *Sales* and *Store Inventory* data marts of Example 4 could retrieve weekly but not daily data. As an example of the latter kind of anomaly, the drill across query over the *Sales* and *Warehouse Inventory* data marts of Example 4 retrieves only data about food and beverage products along years 2002-2003 (see Example 1).

Both cases however refer to a loss of information that is not present in one of the original data marts. Therefore, these anomalies can be tolerated since they correspond to the generation of incomplete but correct results.

To understand the impact of dimension compatibility in drill across queries, let us consider a drill across query involving two comparable but incompatible dimensions d_1 and d_2. According to our definitions, this means that we are able to find data in common between d_1 and d_2, but the operations required to select these data produce some loss in the original dimensions that prevent the correctness of the result of the drill across query. More precisely, incompatibility implies that an aggregation over the original data marts can differ from the computation of the same aggregation over the result of the drill across query.

Example 5. Consider two data marts, one representing the costs of buying a set of products and the second the incomes in selling another set of products. Assume also that the two sets of products are different but overlapping. If we drill across over the two data marts, data is meaningful only for the common products. If we aggregate this data at, e.g., the category level, the costs and incomes obtained for each category are different from those that can be computed over the two individual data marts. □

It is clear that this situation cannot be accepted in drill across queries over autonomous data marts. Dimension compatibility allows us to prevent this anomaly.

We now briefly discuss a problem related to measure (fact) compatibility. Intuitively, two measures of different data marts are compatible when their values can be combined (e.g., compared, added, or multiplied) in some meaningful way. This is very important in drill across queries. However, the characterization of measure compatibility requires a deep understanding of the semantics of measures and aggregate functions, as shown by the following example.

Example 6. Consider the second drill across query proposed in Example 4, over the *Sales* and *Store Inventory* data marts. Their dimensions *Store* and *Warehouse* are compatible at the *city* level. However, joining the two data marts on this common level is meaningful only if a business rule states that each warehouse in a city supply all and only stores in the same city. If this is not the case, the drill across over the *city* level is not correct, since the result of the query will contain meaningless facts. One possible solution could be to exclude the dimensions *Store* and *Warehouse* from the drill across query. □

It is worth noting that dimension compatibility is an extension of Kimball's dimension conformity [6], since dimension conformity implies dimension compatibility, but the converse does not hold in general. We also believe that, in autonomous data mart integration, dimension compatibility can be achieved more often than dimension conformity, and therefore it should be considered the most common criteria when integrating autonomous data marts.

6 Conclusive remarks

In this paper we have investigated the problem of integrating autonomous data marts, with the goal of querying them using drill-across operations. To this aim, we have proposed a novel notion of dimension compatibility and related it to drill-across queries. It turns out the dimension compatibility is a necessary condition to obtain meaningful results.

It should be said that several concepts related to dimension and fact compatibility have been introduced in this paper in a rather informal way. In the future, we plan to study these notions and their relationships with the problem of data mart integration in more depth, both from a theoretical and practical perspective. In particular, we would like to investigate effective strategies for the integration of autonomous data marts.

In this respect, an integration methodology that refers to the notion of conformity could be based around the following main steps.

1. Data marts to be integrated are analyzed, to identify if and how their dimensions are compatible.
2. Semantic matching of compatible dimensions is checked, to verify whether their join is meaningful.
3. Incompatible but related dimensions are identified and, if possible, made compatible on the basis of external information.

Step 1 could be performed using an interactive, semi-automated tool for data mart integration, similar in spirit to Clio [9] and supporting a wide range of scheme-matching and data-mapping techniques. The goal of step 2 is to prevent the combination of compatible but semantically heterogenous dimensions, as discussed in Example 6. Finally, step 3 is oriented towards the identification and enforcement of further matchings and compatibilities, based on inter-scheme knowledge.

External information, if available, should be used in data mart integration, e.g., by adding descriptive data to members. Because of autonomy, such extra knowledge should not be embedded in the original data marts, but it should be rather stored in a ad-hoc repository and managed by a sort of *Federated Data Warehouse System*.

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. 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.
3. L. Cabibbo and R. Torlone. From a procedural to a visual query language for OLAP. In *Int. Conference on Scientific and Statistical Database Management (SSDBM'98)*, pages 74–83, 1998.

4. A. Elmagarmid, M. Rusinkiewicz, and A. Sheth. *Management of Heterogeneous and Autonomous Database Systems.* Morgan Kaufmann, 1999.
5. 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.
6. R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling.* John Wiley & Sons, Second edition, 2002.
7. M. Lenzerini. Data Integration: A Theoretical Perspective. In *21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems*, pages 233-246, 2002.
8. *Microsoft OLE DB programmer's reference and data access SKD.* Microsoft Corporation, 1998.
9. 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.
10. R.J. Miller (editor). Special Issue on Integration Management. *IEEE Bulletin of the Technical Committee on Data Engineering*, 25(3), 2002.
11. 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.
12. 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.
13. E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334-350, 2001.
14. P. Vassiliadis and T.K. Sellis. A Survey of Logical Models for OLAP Databases. *SIGMOD Record*, 28(4):64–69, 1999.