Design and Development of a Tool for Integrating Heterogeneous Data Warehouses

Riccardo Torlone and Ivan Panella

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

Abstract. In this paper we describe the design of a tool supporting the integration of independently developed data warehouses, a problem that arises in several common scenarios. The basic facility of the tool is a test of the validity of a matching between heterogeneous dimensions, according to a number of desirable properties. Two strategies are then provided to perform the actual integration. 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 them. 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 illustrate architecture and functionality of the tool and the underlying techniques that implement the two integration strategies.

1 Introduction

Today, a common practice for building a data warehouse is to develop a series of individual data marts, each of which provides a dimensional view of a single business process. These data marts should be based on common dimensions and facts, but very often they are developed independently, and it turns out that their integration is a difficult task. Indeed, the need for combining autonomous (i.e., independently developed and operated) data marts arises in other common cases. For instance, when different companies get involved in a federated project or when there is the need to combine a proprietary data warehouse with external data, perhaps wrapped from the Web.

We have studied this problem from a conceptual point of view, by introducing the notion of *dimension compatibility* underlying data mart integration [4], which extends an earlier notion proposed by Kimball [7]. Intuitively, two dimensions (belonging to different data marts) are compatible if their common information is consistent. Building on this study, in this paper we illustrate the design of a practical integration tool for multidimensional databases, similar in spirit to other tools supporting heterogeneous data transformation and integration [9].

The basic facility of the tool is the integration of a pair of autonomous dimensions. We have identified 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. The tool makes use of a powerful technique, the *chase of dimensions*, to test for these properties.

Two different integration strategies are supported by the system. The first one 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* [7], based on joining data over common dimension, 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 embeds the sources. Under this approach, queries are performed against the view built from the sources.

The integration of heterogenous databases has been studied in the literature extensively (see, for instance, [5, 8, 12]). 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. Some work has been done on the problem of integrating data marts with external data, stored in various formats: XML [6, 10] and object-oriented [11]. This is related to our tightly coupled approach to integration, in that dimensions are "enriched" with external data. Moreover, our loosely coupled approach to integration is related to the problem of drill-across [1]. However, the goal of these studies is different from ours.

The rest of the paper is organized as follows. In Section 2 we provide the basic notions underlying our approach. In Section 3 we illustrate the techniques for dimension integration and describe how they can be used to integrate autonomous data marts. In Section 4 we present architecture and functionality of the tool and finally, in Section 5, we sketch some conclusions.

2 Matching Autonomous Dimensions

In this section we illustrate the basic issues of dimension matching and provide a fundamental technique, the d-chase, for the management of matchings.

2.1 The framework of reference

We refer to a very general data model for multidimensional databases based on the basic notions of *dimension* and *data mart*. A dimension represents a perspective under which data analysis can be performed and consists of entities

called *members*. Members of a dimension can be the days in a time interval or the products sold by a company. Each dimension is organized into a hierarchy $\preceq$ of *levels*, corresponding to domains grouping dimension members at different granularity. Levels in a product dimension can be the category and the brand of the items sold. The members of the bottom element of a dimension (with respect to $\preceq$) represent real world entities that are called *ground*. Within a dimension, members at different levels are related through a family of *roll-up functions* that map members having a finer grain (e.g., a product) to members having a coarser grain (e.g., a brand) according to $\preceq$. A *data mart* associates *measures* to members of dimensions and is used to represent factual data. As an example, Figure 1 shows a *Sales* data mart that has the quantity, the income and the cost of a sale as measures and is organized along the dimensions *Product*, *Time*, *Store*, and *Promotion*.

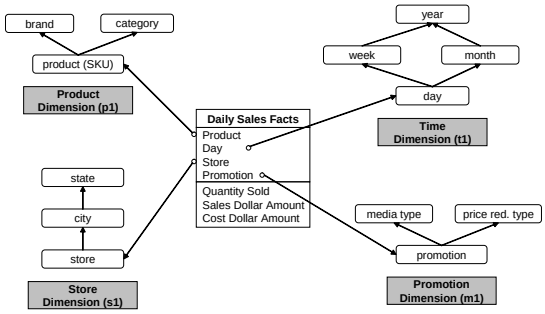

Fig. 1. Sales data mart

2.2 Properties of dimension matchings

The basic problem of the integration of two autonomous data marts is the definition of a *matching* between their dimensions, that is, a (one-to-one) injective partial mapping between the corresponding levels. An example is illustrated in Figure 2, which shows a matching between two heterogeneous geographical dimensions.

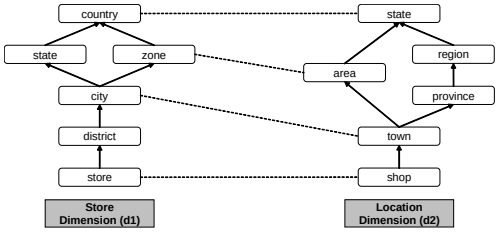

Fig. 2. A matching between two dimensions

We have identified a number of desirable properties that a matching μ between two dimensions d_1 and d_2 should satisfy.

– 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 of d_1 on which μ is defined, there exists a bijection between the members of l and $\mu(l)$;
– Consistency: μ is *consistent* if, for each pair of levels $l \preceq_1 l'$ of d_1 on which μ is defined, the roll-up function from l to l' coincides with the roll-up function from $\mu(l)$ to $\mu(l')$.

A total matching that is coherent, sound and consistent is called a *perfect* matching. Clearly, a perfect matching is very difficult to achieve in practice. In many cases however, autonomous dimensions actually share some information. The goal of the tool we have developed is the identification of this common information to perform drill-across operations between heterogeneous data marts.

2.3 Chase of dimensions

The *d-chase* is a powerful technique inspired by an analogous procedure used for reasoning about dependencies in the relational model [2], which can be used to test for consistency and to combine the content of heterogeneous dimensions. Given a matching μ between two dimensions d_1 and d_2, this procedure takes as input a special *matching* tableau $T_\mu[d_1, d_2]$, built over the members of d_1 and d_2, and generates another tableau that, if possible, satisfies the roll-up functions defined for d_1 and d_2.

A matching tableau $T_\mu[d_1, d_2]$ has a tuple for each ground member m of d_1 and d_2 and includes members associated with m by roll-up functions and possibly *variables* denoting missing information. An example of a matching tableau for the matching between dimensions in Figure 2 is the following.

store	district	city	prov.	region	zone	state	country
1st	v_1	NewYork	v_2	v_3	v_4	NY	USA
2nd	Melrose	LosAng.	v_5	v_6	U2	CA	USA
1er	Marais	Paris	v_7	v_8	E1	v_9	France
1mo	v_{10}	Rome	RM	Lazio	E1	v_{11}	Italy
1st	v_{12}	NewYork	v_{13}	v_{14}	U1	v_{15}	USA
1er	v_{16}	Paris	75	IledeFr	E1	v_{17}	France

In this example, the first three tuples represent members of d_1 and the others members of d_2. Note that a variable occurring in a tableau may represents an unknown value (for instance, in the third row, the region in which the store *1er* 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 value of a tuple t over a level l will be denoted by $t[l]$.

The d-chase modifies values in a matching 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 from l to l' 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.

By applying the d-chase procedure to the matching tableau above we do not encounter contradictions and obtain the following tableau in which, for instance, v_4 has been promoted to U1 and v_{16} to Marais.

store	district	city	prov.	region	zone	state	country
1st	v_1	NewYork	v_2	v_3	U1	NY	USA
2nd	Melrose	LosAng.	v_5	v_6	U2	CA	USA
1er	Marais	Paris	75	IledeFr	E1	v_9	France
1mo	v_{10}	Rome	RM	Lazio	E1	v_{11}	Italy

The d-chase provides an effective way to test for consistency since it is possible to show a matching μ between two dimensions d_1 and d_2 is consistent if and only if the chase $T_\mu[d_1, d_2]$ terminates without encountering contradictions. Moreover, it turns out that if we apply the d-chase procedure over a matching tableau that involves a dimension d and then project the result over the levels of d, we obtain the original instance and, possibly, some additional information that has been identified in the other dimension.

3 Integration techniques

In this section we illustrate two different approaches to the problem of the integration of autonomous data marts.

3.1 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.

We have elaborated an algorithm that generates two expressions of *dimension algebra* that describe, in an abstract way, data manipulations over dimensions [4]. This algorithm is based on three main steps: (i) a test for coherence that takes advantage of the transitivity of $\preceq$, (ii) a test for consistency based on the application of the d-chase, and (iii) the derivation of the selections, projections and aggregations to be performed on the input dimensions in order to select common information.

As an example, the application of this algorithm to the dimension matching reported in Figure 2 returns a pair of expressions that, applied to the original dimensions, generates the dimensions reported on the left hand side of Figure 3.

We have proved that the execution of this algorithm always returns two expressions that correctly compute the intersection of two dimensions if and only if the dimensions are compatible [4].

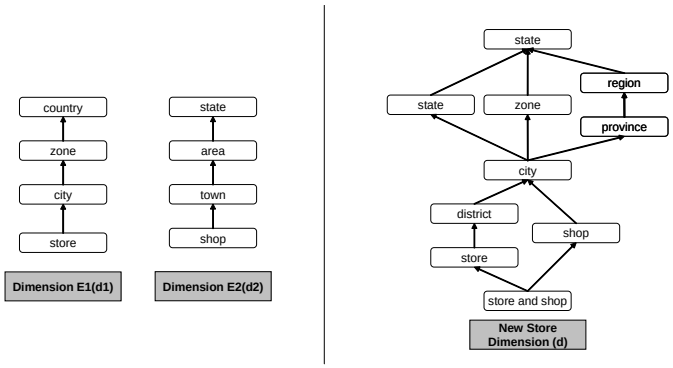

Fig. 3. The dimensions generated by the first algorithm (left) and by the second algorithm (right) on the matching in Figure 2

3.2 A tightly coupled approach

In a *tightly coupled integration*, we want to build a materialized view that combines different data sources and perform queries over this view. 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.

We have elaborated an algorithm that performs this task [4]. This algorithm is also based on three main steps: (i) a test for coherence that takes advantage of the transitivity of $\preceq$, (ii) a test for consistency based on the application of the d-chase, and (iii) the derivation of a new dimension obtained by projecting the result of the d-chase over the "union" of the schemes of the input dimensions. If the union of the schemes produces two minimal levels, the algorithm is more involved since it generates an auxiliary bottom level.

As an example, consider again the matching between dimensions in Figure 2 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. In this case, the application of algorithm to this matching introduces a new bottom level below store and shop. The scheme of the dimension generated by the algorithm is reported on the right hand side of Figure 3.

We have proved that the execution of this algorithm always returns a dimensions d that "embeds" the original dimensions, in the sense that they can be obtained by applying a dimension expression over d [4].

3.3 Data mart integration

Drill-across queries are usually used to combine and correlate data from multiple data marts [7]. These queries are based on joining different data marts over common dimensions and so 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. Assume, for instance, that we wish to correlate the Sales data mart reported in Figure 1 with the data mart storing weather information reported in Figure 4, according to the matchings between the time and the location dimensions as indicated on the right hand side of Figure 4.

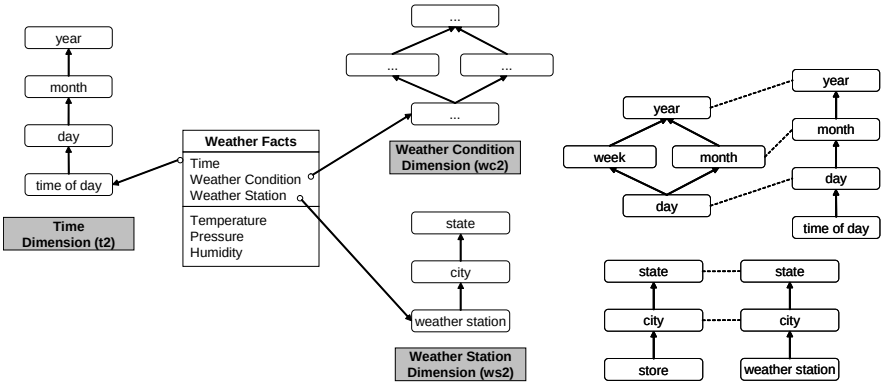

Fig. 4. A weather data mart and a matching between its dimensions and the dimensions of the data mart in Figure 4

The application of algorithm illustrated in the previous section to this input checks for compatibility of dimensions and returns two pairs of expressions that select the members in common in the matched dimensions. 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 these 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 5.

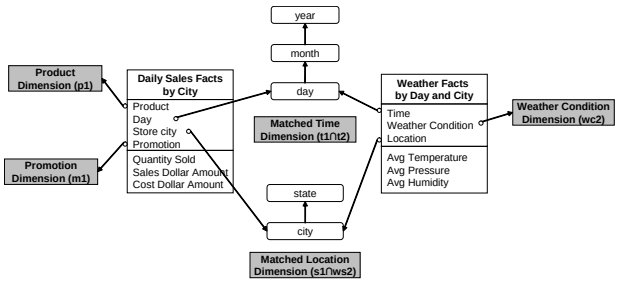

Fig. 5. A virtual view over the Sales and the Weather data marts

The tightly coupled approach aims at combining data from different dimensions by computing their union rather than their intersection. Consider again the example above. If we apply the corresponding algorithm over the same input we obtain two new dimensions that can be materialized and used for both data

marts. Hence, we can then refer to the homogenous scheme reported in Figure 6 to perform drill-across queries.

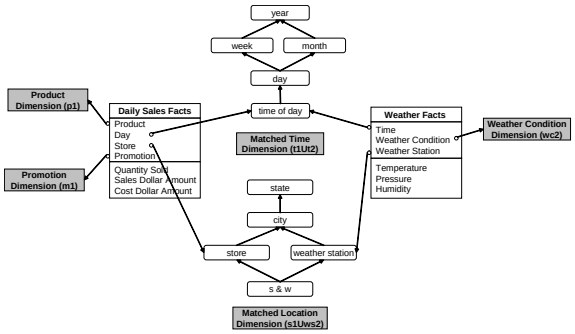

Fig. 6. A materialized view over the merged dimensions

4 The integration tool

The various techniques described in the previous section have been implemented in an interactive tool (screenshots are reported in Figure 7).

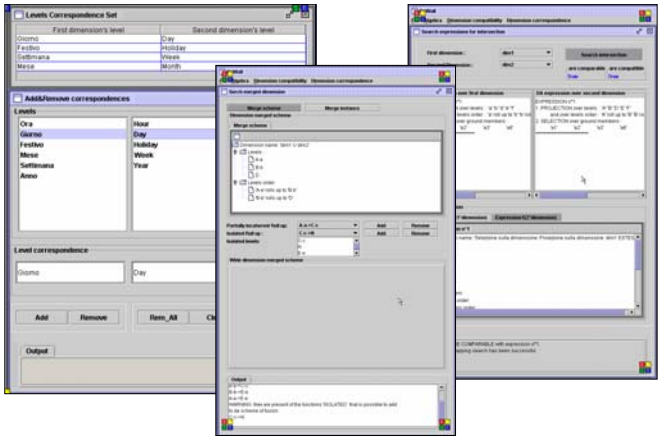

Fig. 7. The integration tool

The tool allows the user to:

1. access to data marts stored in a variety of systems (DB2, Oracle, SQL Server, among others);
2. import from these systems metadata describing cubes and dimensions and translate these descriptions into a uniform internal format;
3. specify matchings between heterogeneous dimensions, by means of a graphical interface;
4. suggest possible matching between levels of heterogeneous dimensions;

5. test for coherence, consistency, and soundness of matchings;
6. generate the intersection of two dimensions, according to the the loosely integration approach;
7. merge two dimensions, according to the tightly integrated approach;
8. perform drill-across queries over heterogeneous data marts whose dimensions have been matched according to either the tightly coupled approach or the loosely coupled one.

Function 4 relies on a number of heuristics that try to infer whether two levels of different dimensions can refer to the same concept. Currently, we have followed a rather simple approach based on the name of the levels and on the existence of shared members. We are currently investigating more involved techniques based on the use of data dictionaries. This is however outside the original goal of our project.

The basic components of the tool architecture are reported in Figure 8.

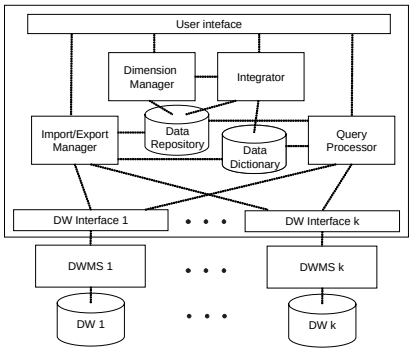

Fig. 8. The architecture of the integration tool

A number of external data warehouses stored in different systems are accessed by the tool through *DW Interfaces* that are able to: (i) extract meta data describing the sources, (ii) translate these descriptions into an internal representation that is based on the multidimensional model described in Section 2, and (iii) store this representation in a data dictionary. The *Dimension Manager* is in charge to specify and verify matching between dimensions. The *Integrator* module performs the actual integrations of pair of dimensions according to the either the loosely coupled approach or the tightly coupled one. In the latter case, a new dimension is built and the corresponding members are stored in a local data repository. 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.

5 Conclusion

We have illustrated in this paper 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 then presented a practical tool that implements the various techniques and can be effectively used to perform drill-across queries between heterogeneous data warehouses. The first experimentations demonstrate the effectiveness of the approach and show a reasonable efficiency. We are currently working to further improve the performance of the tool.

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 (e.g., an *ontology*. This is subject of current investigation.

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. 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.
4. L. Cabibbo and R. Torlone. Integrating Heterogeneous Multidimensional Databases. In *17th Int. Conference on Scientific and Statistical Database Management (SSDBM'05)*, 2005.
5. A. Elmagarmid, M. Rusinkiewicz, and A. Sheth. *Management of Heterogeneous and Autonomous Database Systems*. Morgan Kaufmann, 1999.
6. 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.
7. R. Kimball and M. Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons, Second edition, 2002.
8. M. Lenzerini. Data Integration: A Theoretical Perspective. In *21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems*, pages 233-246, 2002.
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. X. Yin and T.B. Pedersen. Evaluating XML-extended OLAP queries based on a physical algebra. In *ACM Int. Workshop on Data Warehousing and OLAP (DOLAP'04)*, pages 73–82, 2004
11. 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.
12. E. Rahm and P.A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10(4):334-350, 2001.